NASA Technical Memorandum 100693

The Crustal Dynamics Intelligent User Interface Anthology

(NASA-TM-100693) THE CRUSTAL DYNAMICS INTELLIGENT USER INTERFACE ANTHOLOGY (NASA) 57 p CSCL 09B

N88-16448

Unclas
G3/61 0120480

Nicholas M. Short, Jr., William J. Campbell, Larry H. Roelofs and Scott L. Wattawa

OCTOBER 1987

NASA

NASA Technical Memorandum 100693

The Crustal Dynamics Intelligent User Interface Anthology

Nicholas M. Short, Jr.
William J. Campbell
National Space Science Data Center
Data Management Systems Facility
Goddard Space Flight Center

Larry H. Roelofs
Computer Technology Associates
McLean, Virginia 22102

Scott L. Wattawa
Science Application Research, Inc.
Lanham, Maryland 20706

National Aeronautics
and Space Administration

**Scientific and Technical
Information Division**

1987

ACRONYMS

AI	Artificial Intelligence
ART	Automated Reasoning Tool
ATN	Augmented Transition Network
CRUDDES	CRUstal Dynamics Database Expert System
DBMS	DataBase Management System
DIS	Data Information System
DML	Data Manipulation Language
EOS	Earth Observing System
GKS	Graphics Kernel Systems
GSS	Graphics Software System
IDM	Intelligent Data Management
IUI	Intelligent User Interface
KB	Knowledge Base
NSSDC	National Space Science Data Center
NLQP	Natural Language Query Processor
SAIS	Science and Applications Information System
SLR	Satellite Laser Ranging
SQL	Structured Query Language
VLBI	Very Long Baseline Interferometry

CONTENTS

Page

Introduction v

The Crustal Dynamics Project 1

The Crustal Database Information System (DIS) 3

The Relational Model 3
The DIS Database Expertise 8

The IUI Concept 11

Overall System Design 13

The M.1 Expert Systems Tool 15

The CRUDDES Knowledge Base 19

Applications View 19
Architecture View 21
THEMIS Module on PC/AT 24

The THEMIS System 29

Customization Life-cycle 29
Learning Statements 33
Limitations of Natural Language Query Processor Front Ends 33
Limitations of THEMIS in the Development Environment 38

Graphics Interface 39

Overall Graphics Interface Design 39
Detailed Graphics Design 40

Prototype Conclusions and Future Directions 43

APPENDIX A Meta-Facts and Meta-Propositions A-1

APPENDIX B THEMIS Learning Statements A-3

REFERENCES B-1

INTRODUCTION

A first generation prototype Intelligent User Interface (IUI) has been developed for the Crustal Dynamics Data Information System (DIS) as a part of the National Space Science Data Center's (NSSDC) Intelligent Data Management (IDM) Project. By providing a limited sample of the many uses of the DIS, this prototype is intended to demonstrate the IDM concept of removing the user's burden in the information management of complex systems. More specifically, the project will address the issues of applying graphics, natural language, and expert systems technologies to database interfacing.

The DIS was selected as the operational database system because of its consistent architecture, the IUI developers' close working relationship with the DIS developers, and the problems of large databases exhibited in the DIS. Despite the restricted size of the IUI, the system contains sufficient expertise and architectural knowledge to allow limited access into the DIS from a variety of users (i.e., ranging from the naive to moderately sophisticated user).

In the current environment, a user accesses the DIS through the ORACLE Database Management System (DBMS) using either the Data Manipulation Language (DML), Structured Query Language (SQL), or specialized applications programs created through the host language interface. User access to the system requires not only a familiarity with the Crustal Dynamics Project and application programs but also sufficient technical skills in using the DML. While the former requires an understanding of the programs and project in detail, the latter, despite its flexibility, demands that the user know the syntax of the database's query language (SQL), the table names, field names, and data interrelationships. For small database problems like those presented in textbooks, such a task is relatively easy; however, large databases such as the Crustal DIS require a working knowledge of 144 table names and more than 679 field names as well as an understanding of which fields relate to each other. Using such a large database implies that the user must understand its architecture and syntax. Such a task unnecessarily limits the ability of the casual user to access the entire database and, thus, in effect, denies practical access for the user who lacks training in the database environment.

Therefore, the primary goal of the IUI for the DIS is to:

1) Represent the database at a conceptual level such that the occasional user can obtain desired information from the database, with only his/her limited understanding of its structure, syntax, and contents.
2) Support database operations that will allow the occasional user to function and perform database tasks like an expert.

With this goal in mind, the following Technical Memorandum presents the detailed results of the prototyping of an intelligent user interface, which follows from the earlier paper, "The Development of a Prototype Intelligent User Interface System for NASA's Scientific Database Systems."[1]

PRECEDING PAGE BLANK NOT FILMED

PAGE iv INTENTIONALLY BLANK

The authors and development team would like to thank the following people for their technical support throughout the project:

Dr. N. Short, Sr./622
Dr. H. Frey/622
L. Treinish/634
H. Linder/634
C. Noll/634
M. Gough/SAR, Inc.
S. Mathews/SAR, Inc.
E. Hatfield/TechAid, Inc.

THE CRUSTAL DYNAMICS PROJECT

In order to fully understand the Crustal Dynamics IUI, it is useful to first consider the nature of the Crustal Dynamics Project, and, in particular, the database that has been established for that project.[2]

The overall objectives of the Crustal Dynamics Project are to improve the understanding of (a) Regional Deformation associated with the plate boundary in the western portion of North America, (b) Global Plate Motion for the major plates which border or are part of the North America-Pacific pair, (c) Plate Stability for these same plates, (d) Polar Motion and Earth Rotation, and (e) Regional Deformation in other areas of seismicity, especially at plate boundaries. This is done by determining the crustal motions between plates, within plates, and along plate boundaries using two techniques. Both measure baseline and orientation data for a number of observing sites, at an accuracy of about 1 cm in length over thousands of kilometers, and a velocity accuracy of about 1 cm/yr over several years of observation (depending on the plate velocities involved).

The two techniques used are satellite laser ranging (SLR) and very long baseline interferometry (VLBI). In the SLR technique, a pulse of laser light is transmitted from a ground station (either fixed or mobile) to a retroreflector in Earth orbit. Usually, the Moon or the geodetic satellite LAGEOS is the target. By measuring the round-trip travel time for the pulse and knowing accurately the orbital dynamics of the satellite, it is possible to locate the station on the surface of the Earth in an Earth-centered coordinate system. Observations from many stations permit the calculation of the baselines between these stations. By making similar observations over extended periods of time, it is possible to determine the change in such a baseline length. If those stations are on different plates or on either side of an active fault, the changing baseline will measure the plate motion or crustal deformation that is occurring (See Figure 1a).

The VLBI technique is different in that it uses the correlation of radio signals received from extremely distant radio sources (quasars) to determine the baseline between stations observing simultaneously (no such simultaneity is required for SLR because all observations are connected through the satellite orbit). The delay in arrival of a signal from the same source at one station with respect to another depends on the distance between them and the particular observing geometry. By observing a number of radio sources over a protracted period of time it is possible to determine the geometry of the observing stations or network with respect to an assumed fixed station. Figure 1b illustrates this schematically, although in reality the quasars are so distant that parallel radio waves arrive at the two stations. As with SLR, the approach is to repeat the same network geometry changes; these changes can be described as movement of stations with respect to one another and therefore to crustal motion.

For both SLR and VLBI a network of fixed and mobile stations is available. Mobile stations may occupy a large number of sites each year; reoccupation of these sites requires careful local measurements of the actual position of the observing system with respect to the site marker. Fixed sites are "self-marking" in that they do not (in principle) move. Any change to the equipment at a site which may result in a repositioning of the reference observing point must be carefully noted however, in that the measurements being made are in principle accurate at the 1 cm level.

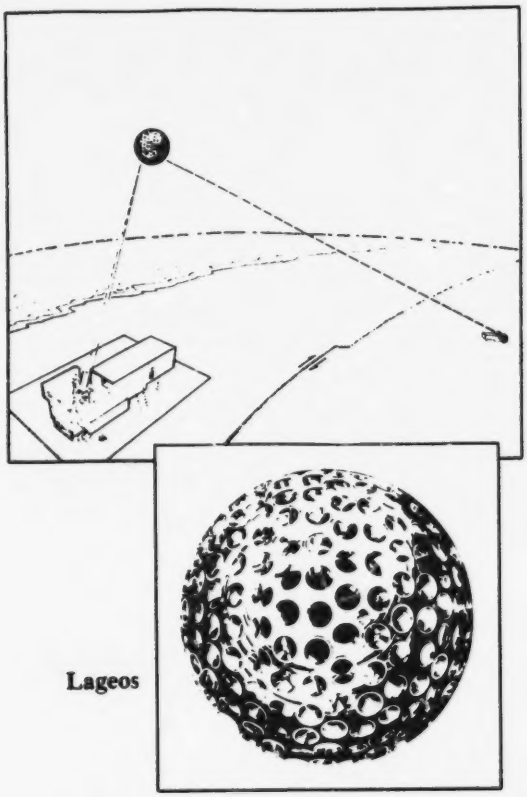


Figure 1a. Satellite Laser Ranging

ORIGINAL PAGE IS
OE POOR QUALITY

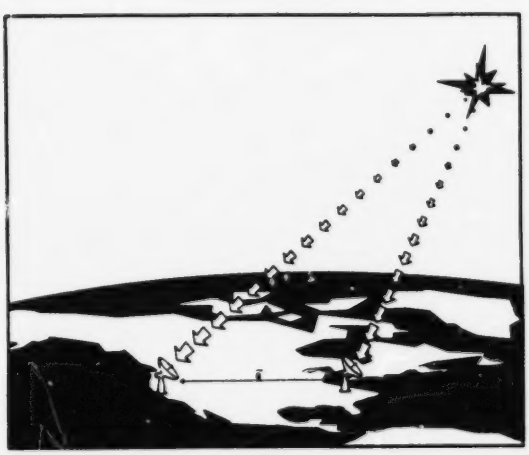


Figure 1b. Very Long Baseline Interferometry

THE CRUSTAL DATA INFORMATION SYSTEM (DIS)

The primary purpose of the DIS is to store, display, and disseminate all geodetic data products acquired by the project in a central data bank and to maintain information about the archival of all project related data (see Figure 2). While the DIS provides several information retrieval services (i.e., Help, SQL, Database handler, Bulletin, News, Data exchange, Report, Screen Forms), the SQL DML option allows the user to access the DIS data in the most flexible manner. Through the ORACLE DBMS, which is based on the 'relational' model, SQL provides access capability to the four general data sets listed below (see Figure 3).

Pre-processed Data - Catalogues of pre-processed SLR from 1976 through 1985. Summaries of SLR data from LAGEOS, BEC, and STARLETTE satellites are stored on-line in a database; the actual data is archived off-line on magnetic tape. The VLBI data consists of on-line experiment listings in the database and a magnetic tape archive of the actual experiment data.

Analyzed Data - SLR, Lunar Laser Ranging (LLR), and the VLBI analyzed results supplied by the Project's Science Support Groups and other analysis centers and Project investigators at GSFC, JPL, NGS, MIT, the University of Texas, and many other global institutions. These analyzed results currently span different periods through the 1976-1984 time-frame and are accessible through the database management system. They include precision baseline distances, Earth rotation and polar motion determinations, length-of-day values, and calculated station positions.

Ancillary Data - This information includes descriptions of Crustal Dynamics Project site locations, a priori monument coordinates and calibration data, and a priori star coordinates. These data sets are contained in the on-line database.

Project Management Information - This category is accessible through the DIS database to authorized Project personnel only and includes mobile system schedules, occupation information, and configuration control information. In addition, DIS operational information is also kept in the database and are accessible to DIS staff only. They include logs of all laser and VLBI tapes created by the DIS for outside users. Listings of DIS back-up tapes are also retained.

THE RELATIONAL MODEL

Before examining the Geodynamics or Crustal database, some notation for the Data Manipulation Language (DML) in the relational model will be described to assist readers unfamiliar with relational algebra.[3] Readers with prior experience in relational systems may refer to Figure 4 for a summary of the algebraic notation used in this paper.

In the Crustal DIS domain, suppose information about the baselines for some particular stations were needed by a user in the database. By formulating and sending a query, the information in Figure 5 is presented to the user. In the database argot, the set of columns (i.e., station, plate, etc.) is referred to as a *relation*; the column name of the relation is called an *attribute* of that relation; and the row of the relation (e.g., 7210, Haleakala, Pacific, 7833, Eurasian, Kootwijk, 10160856.88, etc.) is named a *tuple* of the relation.

Because most relations represent a semantic reference to some purpose, each relation must have a field(s), or an attribute(s), that uniquely defines that relation. For example, the Site_Name attribute in the relation

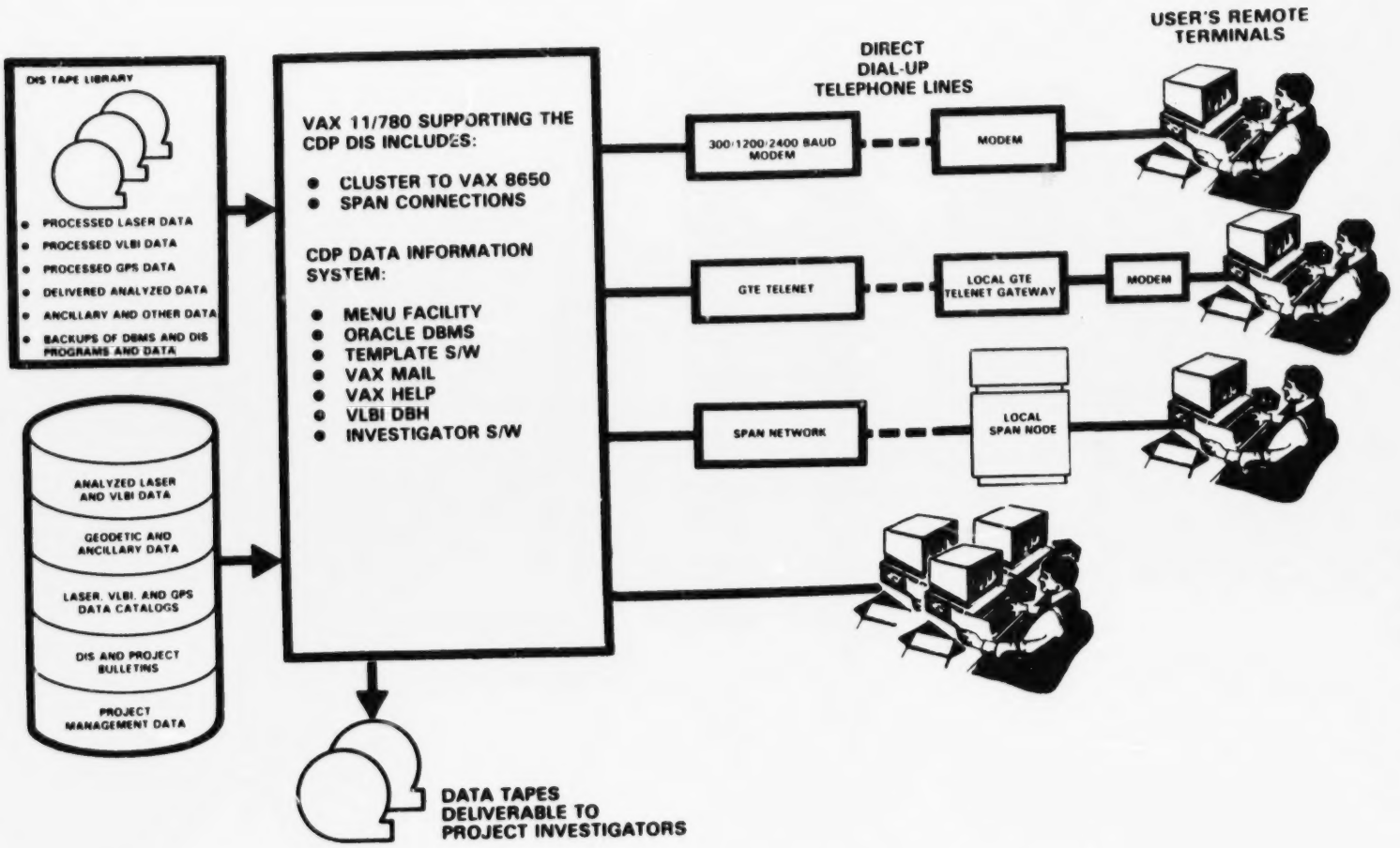


Figure 2. Crustal Dynamics Project Data Information System

ORIGINAL PAGE IS
OE POOR QUALITY

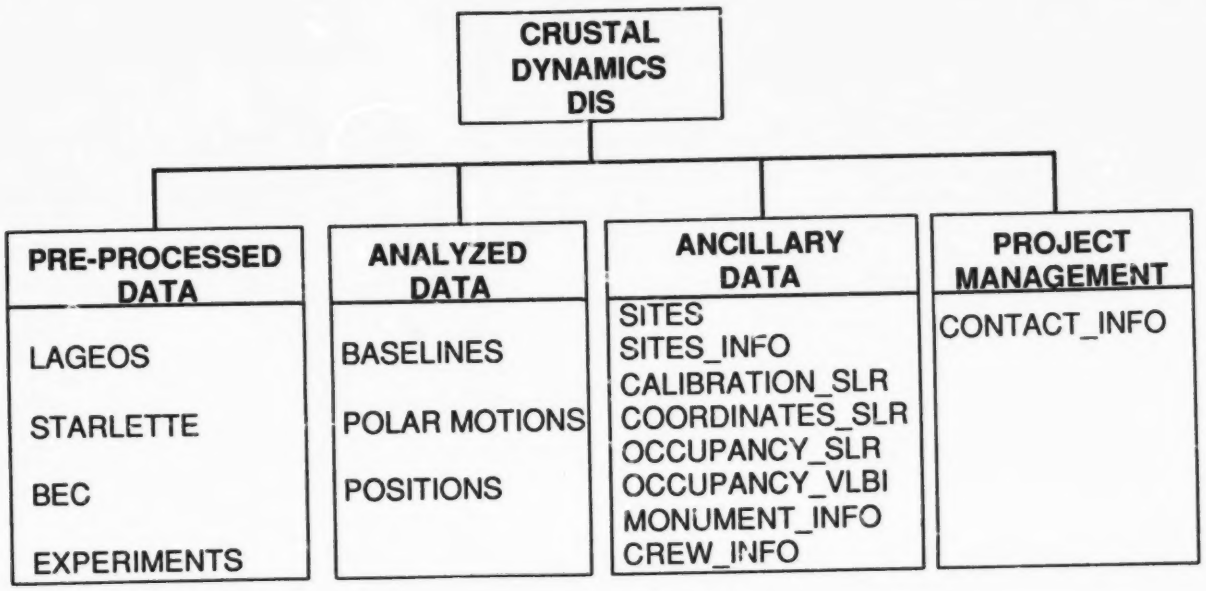


Figure 3. The Crustal Dynamics DIS Table Classes

TOKEN	TYPE	INTENSION
σ	Select	gets specified tuples from a specific relation.
π	Project	gets specific attributes from a specific relation.
$\bowtie$	Join	creates a relation from two relations consisting of all concatenated pairs of tuples such that each pair of tuples satisfy some specific condition.
X	Cartesian Product	creates a relation from two relations consisting of all possible concatenated pairs of tuples, one from each of the relations

Figure 4. Relational Algebra

SITE_LOC

SITE_NAME	PLATE
GSFC	NORTH AMERICAN
HALEAKALA	PACIFIC
VERNAL	NORTH AMERICAN

is a primary key since given site = 'GSFC', no value other than plate = 'NORTH AMERICAN' can exist. On the other hand, the plate attribute is not a primary key because a tuple with plate = 'NORTH AMERICAN' can have more than one value for site (i.e., site = 'GSFC' or 'VERNAL'). Strictly speaking, plate is said to be *functionally dependent* on site because each value of site has exactly one value of plate associated with it at one time.

A relational database can, therefore, be described as the set R ={ all r such that r is a relation }. Along with R, several closed operations exist such that R x R —> R, where x is any of the defined operations. To select a set of tuples from a relation, r, based on some condition, c, the associated algebraic equation is $\sigma_c^{(r)}$.

In our example,

$$\sigma^{(\text{Site_loc})}_{\text{Plate = 'North American'}}$$

produces

SITE_LOC

SITE_NAME	PLATE
GSFC	NORTH AMERICAN
VERNAL	NORTH AMERICAN

While a selection operation chooses rows, the projection operation subsets the columns of a relation and is denoted by

$$\pi^{(r)}_{A_1,\ A_2 \ldots A_n}$$

where A_1, A_2, A_3,...A_n are attributes of relation r. For example,

$$\pi^{(\text{Site_loc})}_{\text{Plate}}$$

produces

SITE_LOC

PLATE
NORTH AMERICAN
PACIFIC
NORTH AMERICAN

In addition to these unary operations, the binary operators, product and join, create a new relation from two existing relations. The product operator is the Cartesian cross-product, r_1 x r_2 for some relations r_1 and r_2, which concatenates pairs of tuples (See Figure 6). If there are n tuples for r_1 and m tuples for r_2, the Cartesian product of r_1 and r_2 has n x m tuples. Unfortunately , this operation can be extremely expensive for large values of n and m and the result is often meaningless.

A more sensible query involves elements of attributes which are equal. In this case, the set or "domain" of site_name in stations_in_sites is a subset of site_name in site_loc. Note that, an attribute A, of relation X is a *foreign key* to an attribute B, the primary key, of relation Y if and only if the domain of A is a subset of the domain of B.

If a selection were performed on r = (site_loc X station_on_sites) where each value of site_name in site_loc equals each site_name value in stations_on_sites, the circled tuples in Figure 6 would result in an equijoin, natural-join, or theta-join.[3] This is denoted by

STATION	CUR_NAME	PLATE	STATION	PLATE	CUR_NAME	BASELINE	CHORD	GEODESIC
7210	HALEAKALA	PACIFIC	7833	EURASIAN	KOOTWIJK	10160856.88	10158350.05	11744947
7210	HALEAKALA	PACIFIC	7834	EURASIAN	WETTZELL	10426979.81	10426979.81	12193219

Figure 5. Relational Output Information

$$r_1 \bowtie_{A_i = B_i} r_2$$

for some relations r_1, r_2 and attributes A_i in r_1 and B_i in r_2. Although a join is a more general form of an equijoin, it will, henceforth, be referred to as just a join .

Of course, because of the natural closure of operations, composite operations such as $(\sigma_c (\pi_A (r))$ are perfectly acceptable and, in fact, are invertible operations.

At this point, enough terminology has been developed to describe the DML in relational algebraic terms. The DML is the set of operations

$\{\sigma_c, \pi, \times, \bowtie$ Union, Intersection, Difference, and Divide$\}$[3]

with respect to the database set R. A DML statement is, therefore, an algebraic expression in the form of an operation or composite operation on R. For example, the SQL statement

```
Select site_loc.site_name, plate, station
From site_loc, stations_on_sites
Where site_loc.site_name =
    stations_on_sites.site_name and plate =
    'NORTH AMERICAN'
```

may be thought of as

$$\left(\pi_{\text{site_name, plate, station}}\left(\sigma_{\text{plate = 'North American'}}\right.\right.$$
$$\left.\left.(\text{site_loc} \bowtie \text{stations_on_site})\right)\right).$$

THE DIS DATABASE EXPERTISE

Although quite flexible, for effective use, the DML in this database requires a substantial amount of database experience because, as previously mentioned, it contains more than 144 table names, etc. For example, suppose one wanted to find all the baselines in the California region along with their location names at each site. The answer for such a question requires that the following sub-questions be answered:

1) Does the user want SLR or VLBI data?
2) Over what time range or year does the user want?
3) At what process location does the user want the data?
4) Does the user wish to use default values for the questions above?

Given the answers to the above, the following example SQL query may result:

```
SELECT F_STATION, S_STATION,
    BASELINE, CHORD, GEODESEC,
    REGION, LOCATION
FROM SITES, BASELINE81_SLRGSFC
WHERE REGION = "CALIFORNIA" AND
    F_STATION = STATION OR
    S_STATION = STATION;
```

While such a query is not complex in SQL terms, the user must know the following knowledge:

1) the chord and geodesic fields are needed in addition to the baseline field;
2) the BASELINE81_SLRGSFC is the mnemonic for the table of 1981 baseline SLR data processed at GSFC and contains the fields in 1;
3) the SITES table must be joined with BASELINE81_SLRGSFC;
4) the fields F_STATION and S_STATION in BASELINE81_SLRGSFC are foreign keys to STATION in SITES.

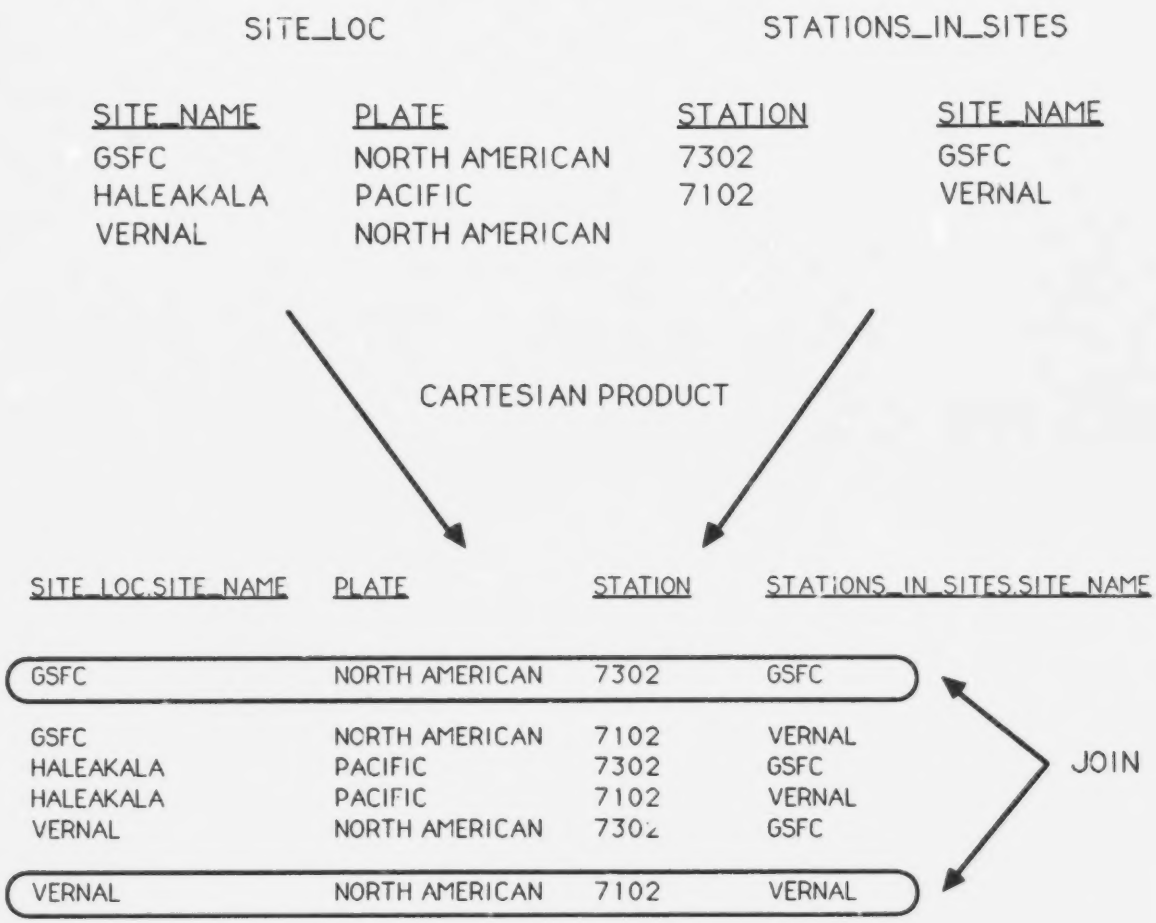


Figure 6. Cartesian Product and Join

Predicated on the above example, it is clear that a large working knowledge of the semantics and syntax of the DML and database structure is needed for even a simple query yet alone more sophisticated access.

Thus, the overall knowledge engineering goal in development of the Crustal IUI involves capturing the database expertise in the form of an expert system (ES) and natural language front-end to the DIS.

IUI CONCEPT

Even though a requirement for knowledge about the database seems obvious, several issues arise about its content and placement within the system.

To begin with, an argument exists for incorporating all the knowledge into the natural language component. Such a premise assumes, however, that in order to retrieve data, the user may only require his own terminology. For example, given the relation

SITE = (Current_Name, Location,
Site_no, Region, Plate)

one could easily formulate the English query

"What are the Site numbers for the North American plate?"

Implicit in such a query is a need for a conceptual model in the user's mind of sites that are identified by numbers and that are located on a particular plate. However, when presented with a relation, such as in Figure 7, conceptual models may fail for most users not familiar with the precise meaning and association of every attribute in the database. The limitations of these natural language front-ends intimates a need for a separation between communicating knowledge and manipulating knowledge. The latter ability suggests an expert system that could manipulate all the precise knowledge in an imprecise manner.

Predicated on the above issues, knowledge in the Crustal IUI could be separated into three categories: *heuristic knowledge*, *virtual objects*, and *procedural knowledge*. Heuristic knowledge, or "rules-of-thumb," corresponds to background information about a specific application which can expedite a search for data (i.e., reduce the search space) [4]. In the DIS database, for example, if the user wants information about plate tectonics, the database can

Normalpt_Ilrtexas

FIELD NAME	FIELD DESCRIPTION
INDX	Index for joining with ancillary data
OBS_DATE	Observation date
EPOCH	Observation epoch
RC	Reflector code
ETB	Epoch time base
TDELAY	Observed time delay
UNCER	Uncertainty estimate
EDELAY	Electronic delay
GDELAY	Geometric delay
NUMSTOPS	Number of photon stops in normal points
FRQOFFSET	Frequency offset
DTB	Delay time base
PRESS	Atmospheric pressure

Figure 7. Example of a Relational Table

offer data about regional deformation, plate stability, or motion of the whole plate.

The second class of knowledge, virtual objects, arises when discussing, for example, plate stability. The object, plate stability, is equivalent to a selection of all the baselines on a specific plate. While "plate stability" does not exist as an actual relation or attribute in the database, it represents a selection on a projection of the class of baseline tables combined with the site table. A virtual object could, therefore, be defined as:

- an inferred object resulting from the cluster of either tuples or attributes or
- a synonym for a virtual or a real data object.[1]

Of course, this clustering operation implies a selection or projection on some relation(s).

But, theoretically, any operation requiring a procedure could operate on either real or virtual objects in order to create new objects. For example, relative plate motion is determined by iteratively taking the baselines between the respective plates and applying an algorithm similar to a least square fit on the baselines. The result is a new attribute or virtual object of motion vectors. This knowledge of the algorithm corresponds to the aforementioned category, procedural knowledge.

Just as SQL relations may be organized into particular (application) views, these types of knowledge exist in organized views corresponding to classes of users.[3],[1] These different classes result from the differences among the sets of user's use of virtual objects and heuristics. Referred to as *relativism*, this ability to perceive a segment of information in different ways based on the users point of view requires a "polymorphous representation" that is not adequately utilized in the relational model.[10] For example, in order to determine plate stability, a scientific user may be interested in Site-1 for its location on a plate whereas an engineer may only consider Site-1 for calibration needs. Thus, the role that Site-1 partakes in the two virtual objects, plate stability and calibration information, depends on the view or context of the user's situation. In this manner, resolution of context amounts to a determination of the type of user (i.e., scientist, engineer, project manager, etc.). As with any classifi-cation process, however, a problem arises when the user does not fit one of the stereo-types. In such an event, a default view representing a logical taxonomy of the entire database (Architecture view) must exist to permit immediate branching from one view to another.[1]

Thus, the combination of the two concepts, knowledge type and views, in an IUI offers the following advantages:

- It can facilitate understanding by specifying the contents and meanings of a collection of data as well as the relation between objects within the data;
- It can support approximate reasoning to infer conclusions that are not explicitly stated by the user, such as an imprecise or fuzzy query which can be stated without mentioning file names or table joins;
- It can handle fuzzy concepts, expressed by using a fuzzy query;
- It can handle information demands for which the database is used routinely, rapidly, and efficiently with little understanding by the user;
- It can communicate with the user in English text;
- It can serve as the semantic basis for understanding the user's data needs in conjunction with the natural language front-end;
- It can provide a logical representation of the database dictionary or architecture and stored information for the casual user;
- It can facilitate the identification and understanding of database information from the database operational view (i.e., the database design model).[1]

OVERALL SYSTEM DESIGN

In terms of system design, the Crustal IUI consists of an aggregate of several commercial packages. Residing on two different computers, this system utilizes three existing software technologies: graphics, expert systems, and natural language query processing. While the database and DBMS existed before the IUI prototype project, the Expert System (ES) component was created using the M.1 expert system development tool produced and marketed by Teknowledge, Inc. Although M.1 provided the reasoning environment for the ES, the knowledge about the Crustal database has the form of a secondary database-like section called a Knowledge Base (KB).

Like the expert system component, the natural language portion was developed using the commercial package THEMIS (Frey Associates, Inc.). In general, THEMIS consists of a set of parsing algorithms with an English-like grammar and a dictionary of English words that also identifies their parts of speech. In addition to the provided dictionary of common English terms, the Thesaurus was augmented with references to terminology residing in the DIS. This customization of THEMIS must be programmed interactively using "learning statements" (not implying that THEMIS learns automatically). By using plain English, the customized system can retrieve information (See Figure 5) from the Crustal database.

The Crustal IUI environment resides on a combination of an IBM PC/AT and the NSSDC VAX 11/780 (See Figure 8). From the user's perspective, the Crustal database is accessed through either the expert system or THEMIS via the communications package on the PC/AT. In other words, if the user feels comfortable in forming English queries, the user may only need to use THEMIS to access ORACLE tables. Otherwise, he may obtain guidance by using the expert system.

In general, the expert system determines the user's database needs by questioning that user. Unlike many of the expert systems which provide only advice about a problem domain, this expert system, called the Crustal Dynamics Database Expert System (CRUDDES), not only provides the advice but actually retrieves the information by passing an English query to THEMIS (using the communications package).

After an English query is transmitted from the PC/AT workstation to the VAX, the natural language processor, THEMIS, parses the query into a SQL query via an interlingua (i.e., internal representation in THEMIS). Once translated, THEMIS passes the SQL query to ORACLE, which processes and returns the desired information back to the AT in the form of several columns of data (see Figure 5).

The expert system, communications package, and graphics were customized to reside in the PC/AT's main memory, thus, avoiding excessive I/O swapping. M.1 allows external function calls which permit conventional software to be imbedded in the expert system's environment at run-time. Unfortunately, problems occur with this architecture when excessive amounts of dynamic memory are required. In other words, both the expert system and external functions share the same memory space, so a conflict may occur at an inopportune time (e.g., during iterative consultations). These conflicts are exacerbated because once the version of M.1 used allocates space, recovery of the space cannot be properly done (garbage collection).

The current PC/AT component of the system uses 640k RAM, a Hercules monochrome graphics board, and a Mouse Systems mouse. Although system-independence is desired for the software, "in-house" hardware limits the choice of commercial development software.

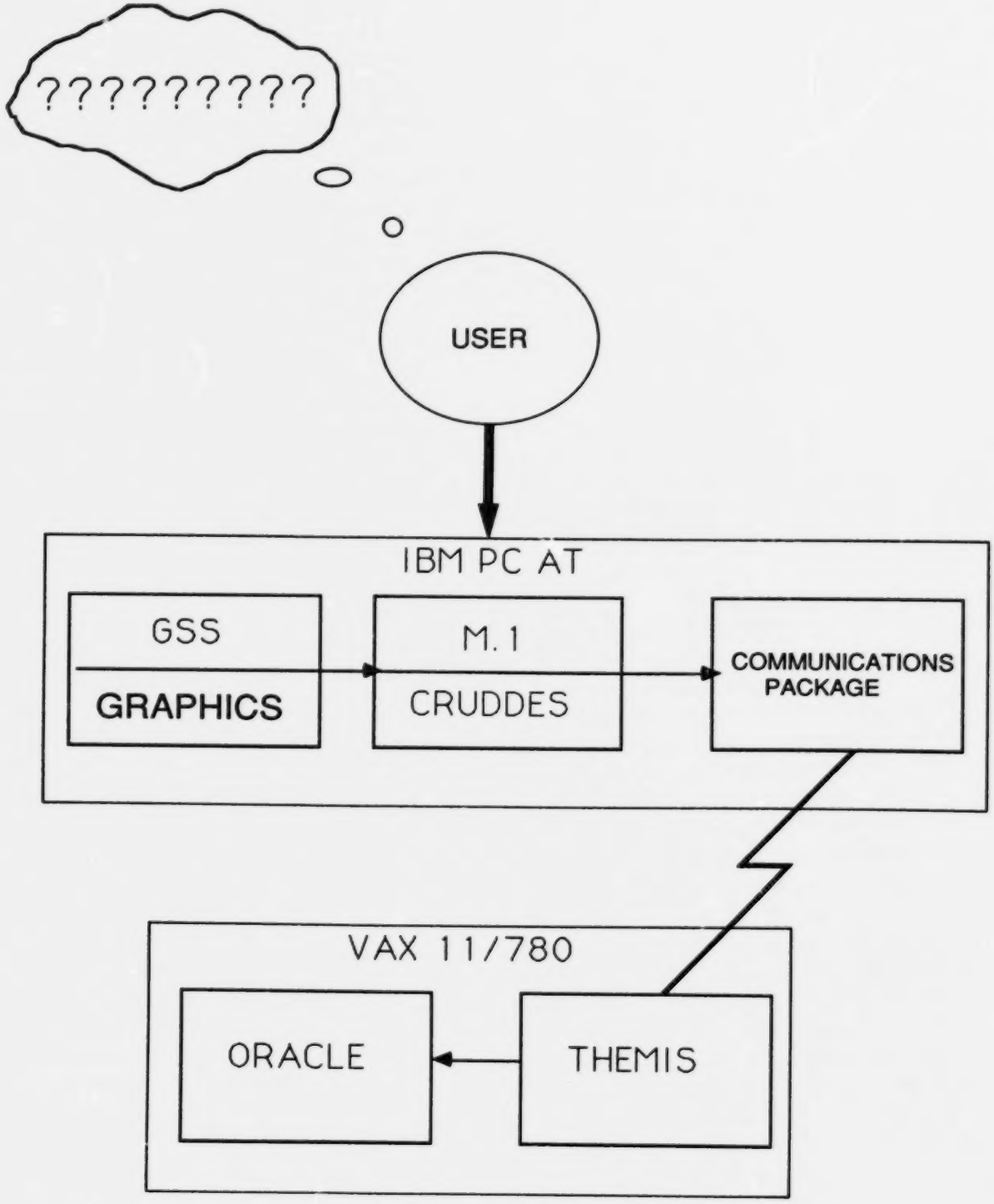


Figure 8. Crustal IUI Environment

THE M.1 EXPERT SYSTEMS TOOL

The tool for creating the expert system, M.1, consists of three basic components: the *inference engine*, the *knowledge base*, and the *cache* (see Figure 9). Basically, the inference engine manipulates the applications-specific KB in order to arrive at the correct conclusions. More specifically, M.1 uses a reasoning paradigm, known as backward chaining, which eliminates needless assertions typical of data-driven (forward chaining) paradigms.

In essence, backward chaining, as opposed to forward chaining, assumes a goal exists. For this goal to be true, a set of sub-goals must also be true in order to propagate conclusions up to the main goal. When these sub-goals are solved, the reasoning process is complete; otherwise, a new goal must be determined and another whole process restarted. Note that the old conclusions determined by the previous process govern the conclusions of subsequent processes. To save search time, the previous conclusions reside internally in a buffer or cache memory. In this manner, the inference engine sets priority by first searching the cache and, then, looking through the KB.

In its rudimentary form, the KB can be considered as a loosely connected set of rules. Basically, these rules typically consist of the IF-THEN knowledge representation scheme. Syntactically, a rule often takes the form of:

```
IF    <antecedent>  1          and/or
          :   :
      <antecedent>  N
THEN  <conclusion>  1          and
          :   :
      <conclusion>  M
```

Notice that, for a series of antecedents, several conclusions can be reached. Regrettably, because of the control structure in the inference engine, M.1 can only conclude several values for one conclusion type. For example, the rule

```
IF      site_type = slr
THEN    site = 34 and
        site = 45
```

is legal, whereas the rule

```
IF      site_type = slr
THEN    site = 34 and
        station = 7301
```

is not. Of particular note, this illegal M.1 expression can be mimicked using a method called a *whenfound* statement. The above could be reformed as

```
IF      site_type = slr
THEN    site = 34.
WHENFOUND ( site = 34 ) =
DO (set station =7301 ).
```

Basically, after M.1 concludes an inference step, it searches the KB for statements called *meta-facts* to produce side-effects (see Appendix A.). These side effects are the vestigial consequences of some action, such as when site is 34, set station to 7301. Of course, a duplicate rule could have been created with the same antecedents, but then some unwanted effects might occur on the flow of control. Probably the best way to elucidate M.1 's backward chaining schema would be to present the following simple example KB:

```
goal = job_is_done.
rule-1:     if      site_type = vlbi
            then    site = 35.
rule-2:     if      site_type = slr
            then    site = 40.
rule-3:     if      site is known
            then    job_is_done = yes.
question(site_type) = 'What type of site is this
(SLR/VLBI)?'.
legalvals(site_type) = [slr, vlbi, unknown].
```

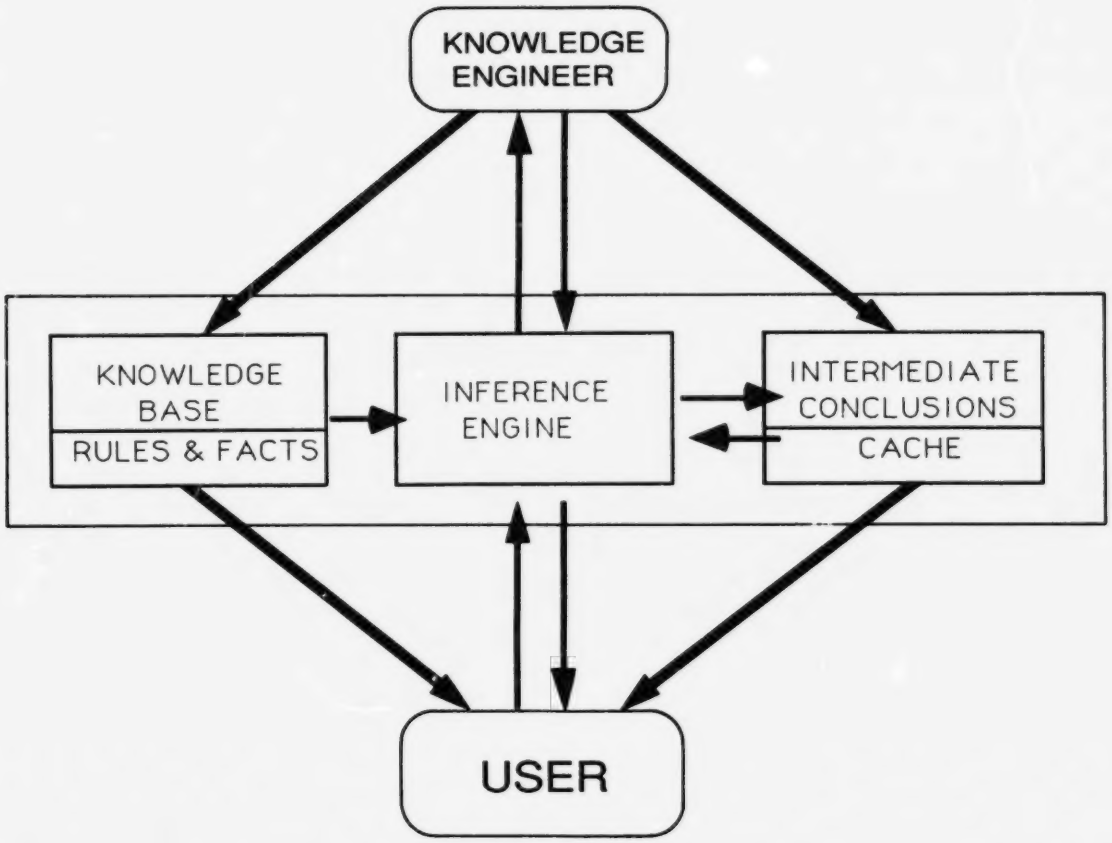


Figure 9. M.1 Architecture

In this example, M.1 's first task involves looking for the values needed to satisfy the initial goal "job_is_done," as stated in the KB entry "goal = [goal-variable]." After searching through the cache, which starts out empty, M.1 searches each conclusion for the variable job_is_done. In this example, M.1 finds the variable, called an expression in the M.1 argot, in rule-3. To conclude job_is_done = yes, M.1 checks if all its antecedents are true. In this case, M.1 checks if "site is known" by first examining the cache and then searching the KB for the first expression site = value. Next, rule-1 is invoked in a similar manner. However, in this case, neither a cache conclusion nor another rule for "site_type" exists for M.1's use. To resolve this, M.1 searches the KB for the question meta-fact statement to cause the question "What type of site is this (slr/vlbi)" to be issued to the user. Finally, the user can respond with either of the legal values [slr , vlbi, unknown] presented in the legalvals meta-fact statement. Of course, if the user responded "vlbi," rule-1 sends site = 35 to the cache, in which case site is known and job_is_done equals yes.

However, if the user responded "slr," rule-1 fails. Inasmuch as another rule exists with the site expression in its conclusion, M.1 invokes the next rule, namely rule-2. With site_type = slr, site = 40 is concluded and the process continues. Of particular importance, therefore, is the ordering of the rules. For example, suppose the user responds "vlbi" 75% of the

time. Then, by placing rule-1 in front of rule-2, M.1 could avoid having to test rule-2 75% of the time. In large KB's, reordering the rules can greatly increase systems performance. Such a concept of control information, by the way, is referred to in Artificial Intelligence (AI) as *meta-knowledge* about the use of the knowledge in the KB. M.1 contains other methods for controlling its reasoning path such as using the aforementioned meta-facts and a type of antecedent called a meta-proposition (see Appendix A. for more details).

THE CRUDDES KNOWLEDGE BASE

Based on the research on user interfaces and database theory [1], the architecture of the Cruddes KB was formulated into two views of the current database, the applications view and the architecture view. For the applications view, only a view of specific interest to the science users was incorporated in the prototype even though project managers, the database administrator, engineers, etc. will use the DIS.

The architecture view was created to act as a default view for cases when:

1) the user realizes that he doesn't know his needs;
2) the user has led the system down the wrong reasoning path by entering inappropriate responses;
3) the user realizes that he doesn't want what the system is providing;
4) the user has several different responsibilities, such as a project manager who is also an engineer;

Since either of the two views or segments may be chosen, a mechanism for linking the two must be in place. Such a mechanism, called a *transform filter*[1], must choose, based on the previously inferred data, which location to branch to in the architecture view. For CRUDDES, only two methods were used due to the limitations of M.1. First, where there existed a specific location to branch, a fixed pointer in the form of a rule was created for branching to the architecture view. Second, where no known pointer location was known, a key-word pattern matcher, shaped after Eliza[5], was used. Basically, when the pattern matcher is on-line, the user is questioned as to his needs and told to respond in plain English. The pattern matcher then looks for key-words in his response to serve as clues for a location in the architecture view. If no match is found, the system defaults to the beginning of the architecture view. If one is found, however, the key-word per se is used to point to the appropriate node. For example, suppose the user typed in "I want calibration information about sites." The system then recognizes "calibration" and uses it to point to the appropriate rule

```
IF      'calibration' and
        ques-14 = a
THEN    objects = 'calibrator'.
```

Based on ques-14 being unknown, the system responds

"Do you want to know
  a) the calibrator
  b) the calibration date
  c) or the table board information?"

In essence, the system is cueing the user that just asking for calibration information alone is insufficient and, in actuality, the user must resolve the *word-sense* ambiguity of calibration by presenting the three types of calibration data. Figure 10 summarizes the logical component level.

Since the approach of the two views is both functionally and stylistically disjointed, each view is discussed separately.

APPLICATIONS VIEW

Figure 11 diagrams the functional hierarchy or rule path of the applications view. This hierarchy closely resembles an "and/or-graph," which is goal-directed.[5] In general, the diagram reflects the scientific purposes presented in Section 2. The whole plate module in the diagram needs further explanation. This module can be subdivided into two sections: entire plate motion and plate motion between regions.

PRECEDING PAGE BLANK NOT FILMED

CRUDDES KNOWLEDGE BASE DESIGN

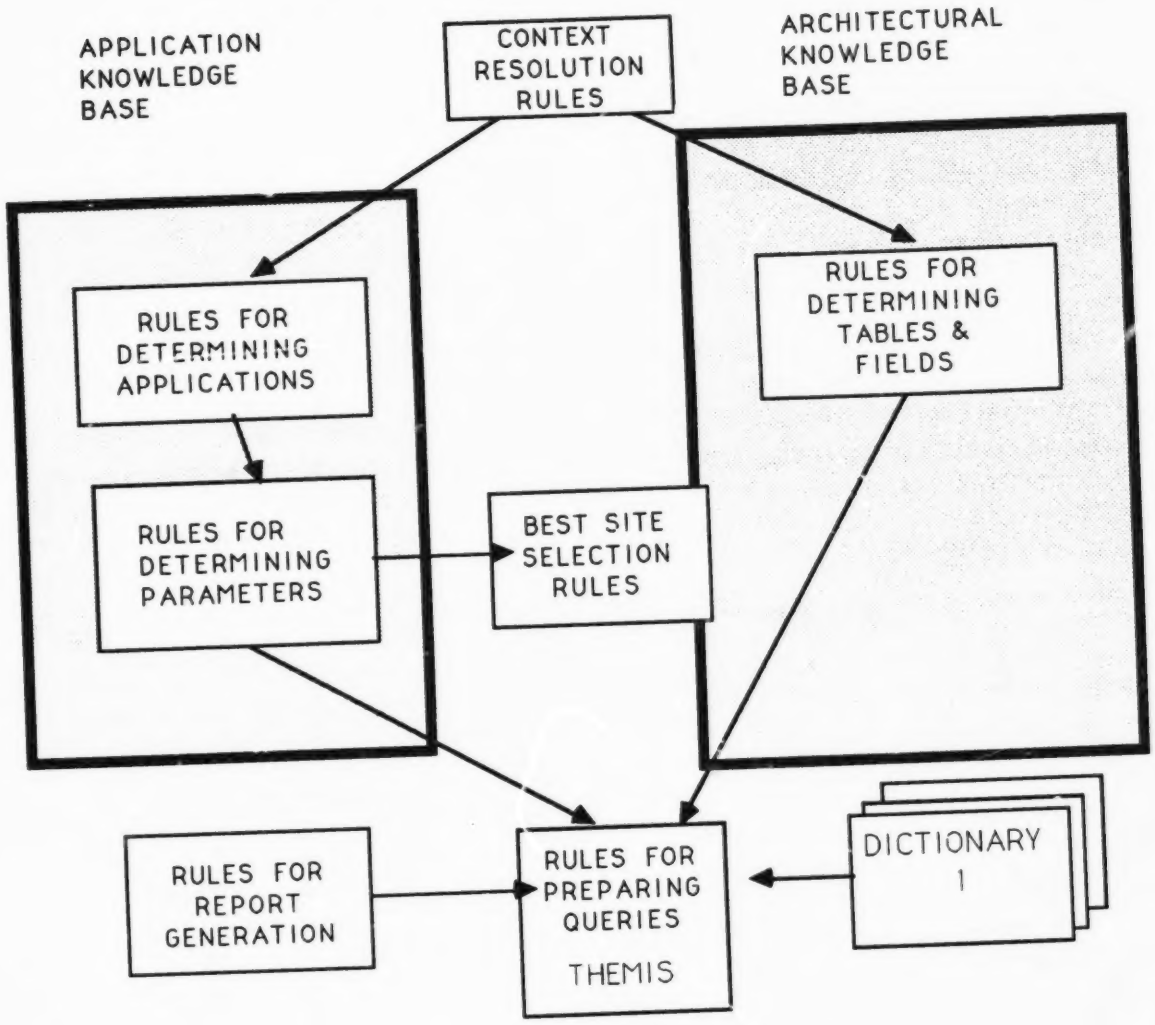


Figure 10. Logical Components of Cruddes

Entire plate motion denotes the case when scientists are interested in an average motion between two plates. To accomplish this in terms of the database all baselines between these plates, determined over a given time span, must be specified. For plate motion between regions, owing to the fact that plates do not necessarily move uniformly internally as well as relative to each other, all the baselines between desired regions must be provided for comparisons of rates between regions on the same plate. For example, suppose the North American plate adjacent to the zone along the Northwest U.S. coast is moving at a different rate than Central America within the same plate, each in relation to the Pacific and Nazca plates. By providing the baselines from these respective regions, the user's goals would be satisfied.

Although not treated earlier in Section 2, the concept for determining the distance between two points is provided by a module within CRUDDES. This module may be considered as a tautology for whole plate motion, except that there are some subtle differences in both purpose and functionality. This module represents one of the fixed transform filters in the architecture view.

From the user's perspective, the system asks the user for specific site numbers, assuming that he may recall the numbers. If the user can't provide such information, the system generates a world map on the CRT display with the sites shown as icons in their respective latitude/longitude positions (see section on graphics). Once sites are chosen by the user, another query calls up the appropriate baselines as a function of parameters such as satellite type, year, process location, etc. The difference in purpose between this module and the three other modules in the plate tectonics component is not only in a better order of the chosen baselines but in the semantics of the reasoning path.

Other considerations in formulating this view involve the use of "straight forward" knowledge engineering techniques. For instance, being able to enter mnemonics such as "w" for "Whole plate motion" enables the user to respond quickly and, thus, with much less typing. Likewise, the use of the explanation meta-fact aids in describing the terminology used or in showing how the system arrived at its previous conclusions. For example,

> explanation (rule-a2) =
> ['The database has two basic types of scientific information accessible to the user, these being either Plate tectonics or Earth rotation information. If the user is unsure of what he wants then an "uncertain response" will invoke the presentation of an architectural view of the database which can be used to identify and determine what information might be of interest']

describes the general purpose of the expert system.

Lastly, because CRUDDES is a "proof of concept" and demonstration system, both the Earth rotation and regional deformation sections remain unfinished. The limited available time was spent developing the user interface paradigm.

ARCHITECTURE VIEW

Unlike the applications view, which tends to search for parameters in a limited domain, the architecture view represents an "Or-graph"[6] tree structure guided by the user's answers (see Figure 12). For example, the rule-base section

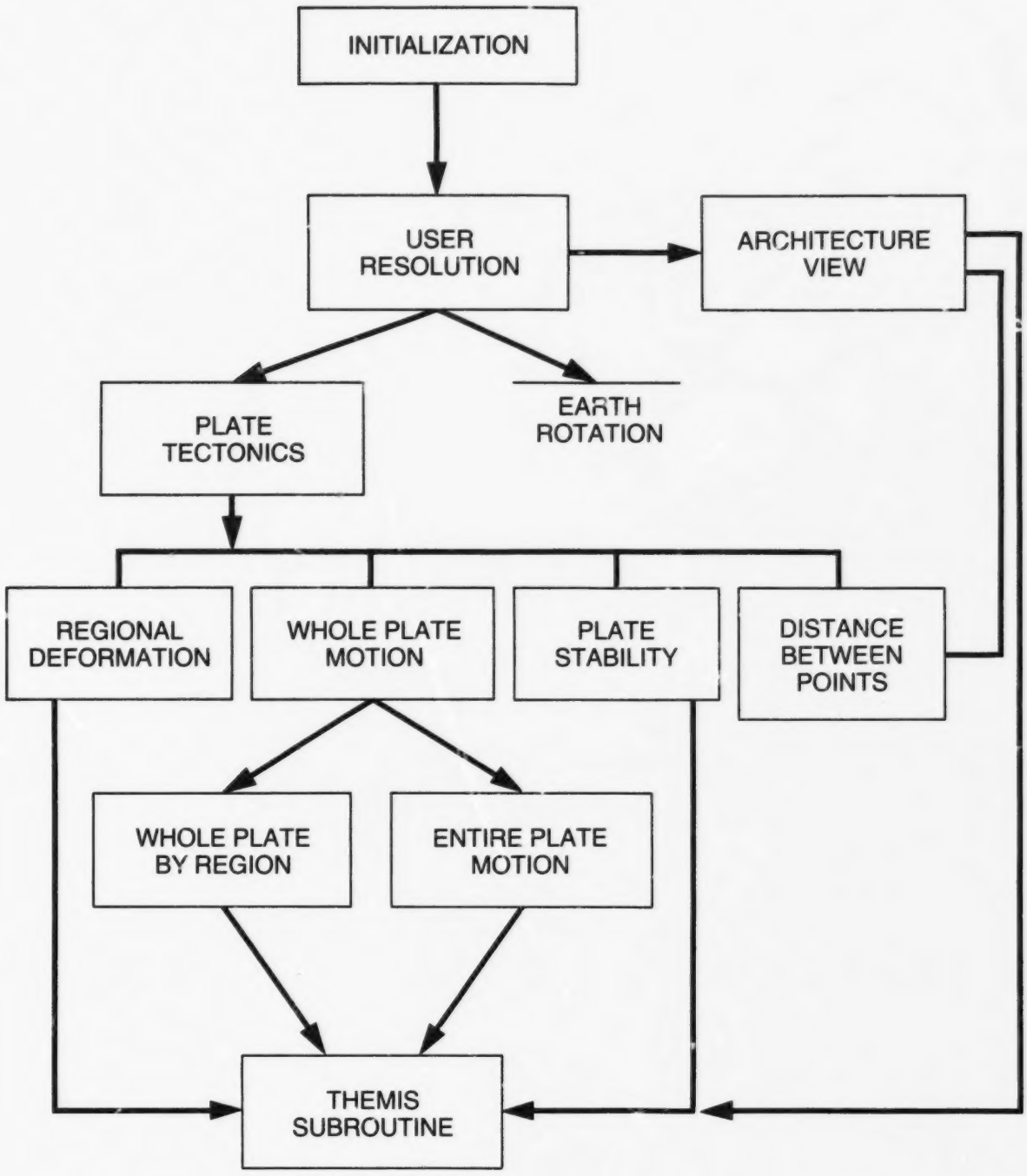


Figure 11. Applications View Functional Hierarchy

```
IF      architecture-view and
        ques-1 = a
THEN    'site specific'.
question(ques-1) = 'Do you want
                    a) site specific
                    b) experiment specific
                       information'.
IF      architecture-view and
        ques-1 = b
THEN    'experiment specific'.
```

shows the top-level search. If the user chooses "b" for ques-1, then 'experiment specific' = yes ("yes" is implied in the expression) brings on the next rule

```
IF      'experiment specific' and
        ques-22 = a
THEN    'results'.
```

to be triggered. Eventually, the search terminates in a rule like

```
IF      'position' and
        ques-12 = a
THEN    objects = 'latitude and longitude'.
```

The value of the last expression, objects, represents one or all of the strings to be instantiated into the English query. In relational algebra terms, the objects act as indexes to relations, projected relations, joined relations, or a combination of all three. Also, the indexes, or

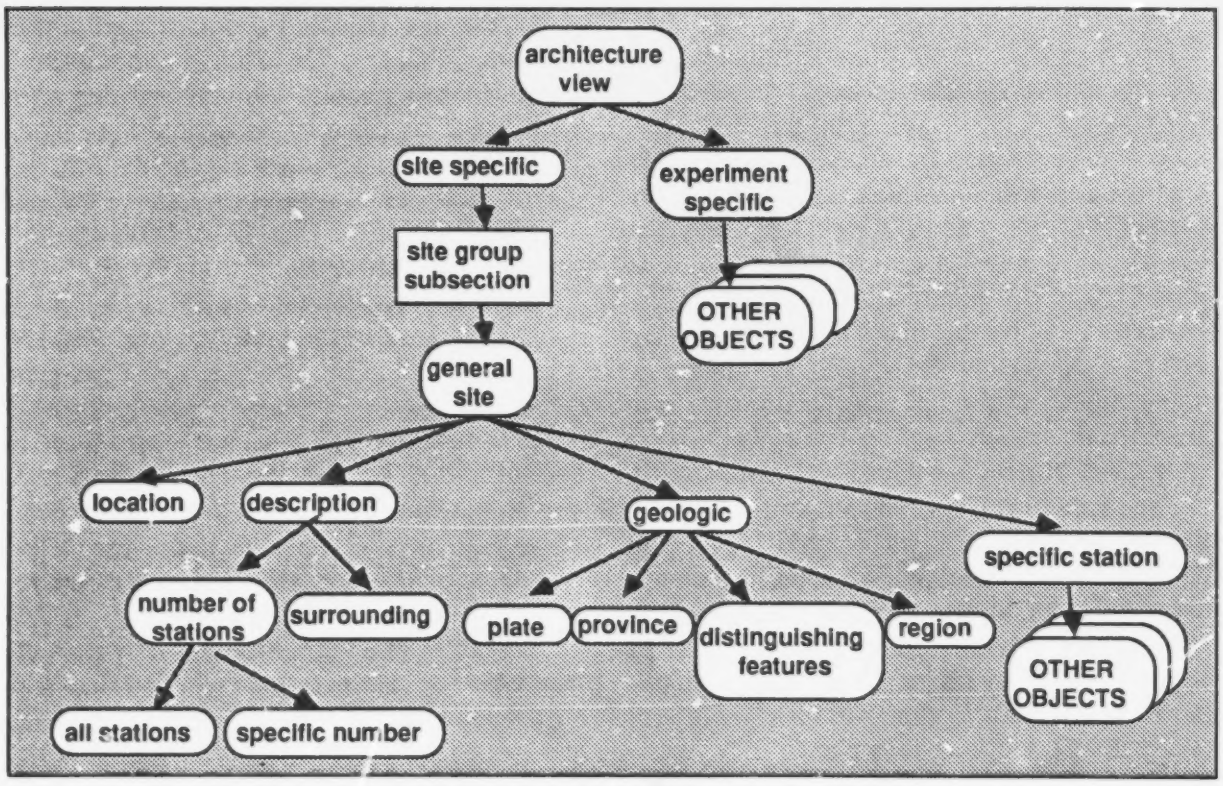


Figure 12. Architectural View

expression values, act as key-words for the pattern-matcher transform filter in the case where the user finds the query through serendipity. For instance, suppose the user states,

> "I want the latitude and longitude of the sites "

then the system would generate the appropriate query response,

> "show me current name, location, latitude, and longitude."

The interesting point about this query lies in the addition of the current name and location fields. If the user asks

> "show me latitude and longitude,"

THEMIS responds by printing meaningless latitude and longitude numbers. In essence, this query is fuzzy because the attributes, latitude and longitude, are functional dependencies on a determinant field(s) that is not included. For this reason, a main function of the system is to provide all the correct fields to thwart the above meaningless responses.

Even though most of the nodes of this view are of a consistent data-type (i.e strings or expressions), one major deviation occurs in the site specific branch (see Figure 13), namely the interactive graphical selection of sites.

Because it is pointless to present specific information about sites without the location of sites, this section was integrated into the view to allow the user to first select the sites before choosing the relevant data. For example, there is no sense in asking for descriptive, location, geologic, or specific station data without first referencing the appropriate sites.

In general, the architecture view functions as a logical taxonomy for the database with data concerning both sites and experiments as the highest abstractable form. Although there is a similarity to a menu-driven system, one major difference lies in the ability to move easily to different levels. For instance, after the "distance between points" branches to the sites section, a branch to the experiments results module takes place as presented in Figure 14. In fact, this ability makes M.1 very amenable, although several costlier expert systems tools contain these dedicated portions automatically.

THEMIS MODULE ON THE PC/AT

This module remains as a vestige from a previous version, which required a segmentation of the knowledge base. Although no longer needed, the original design concept dictated that database objects be determined first by backward chaining. Once the amorphous set is established, a pseudo forward chaining paradigm could be used to instantiate the objects into a English query structure of the form:

```
themis-OBJECT-REF-MOV-LOW-HI-
TYPE = [ ' show me all the ',TYPE, '
baselines between the ',REF, ' ',
OBJECT, ' and the ', MOV,' ', OBJECT,'
during 19 ', LOW, ' and 19 ', HI,' . ',nl ].
```

In M.1 syntax, variables are represented by capital letters so that the above expression contains values that are instantiated via rules. In the architectural view, this instantiation concatenates operated relations (i .e., projected or joined) to the selection operation.

Currently, three sub-modules comprise the major part of this subroutine . The first sub-module reflects the applications view and the above discussion, the second sub-module, the architecture portion, presents, for the most part, two query options. The first query consists of a

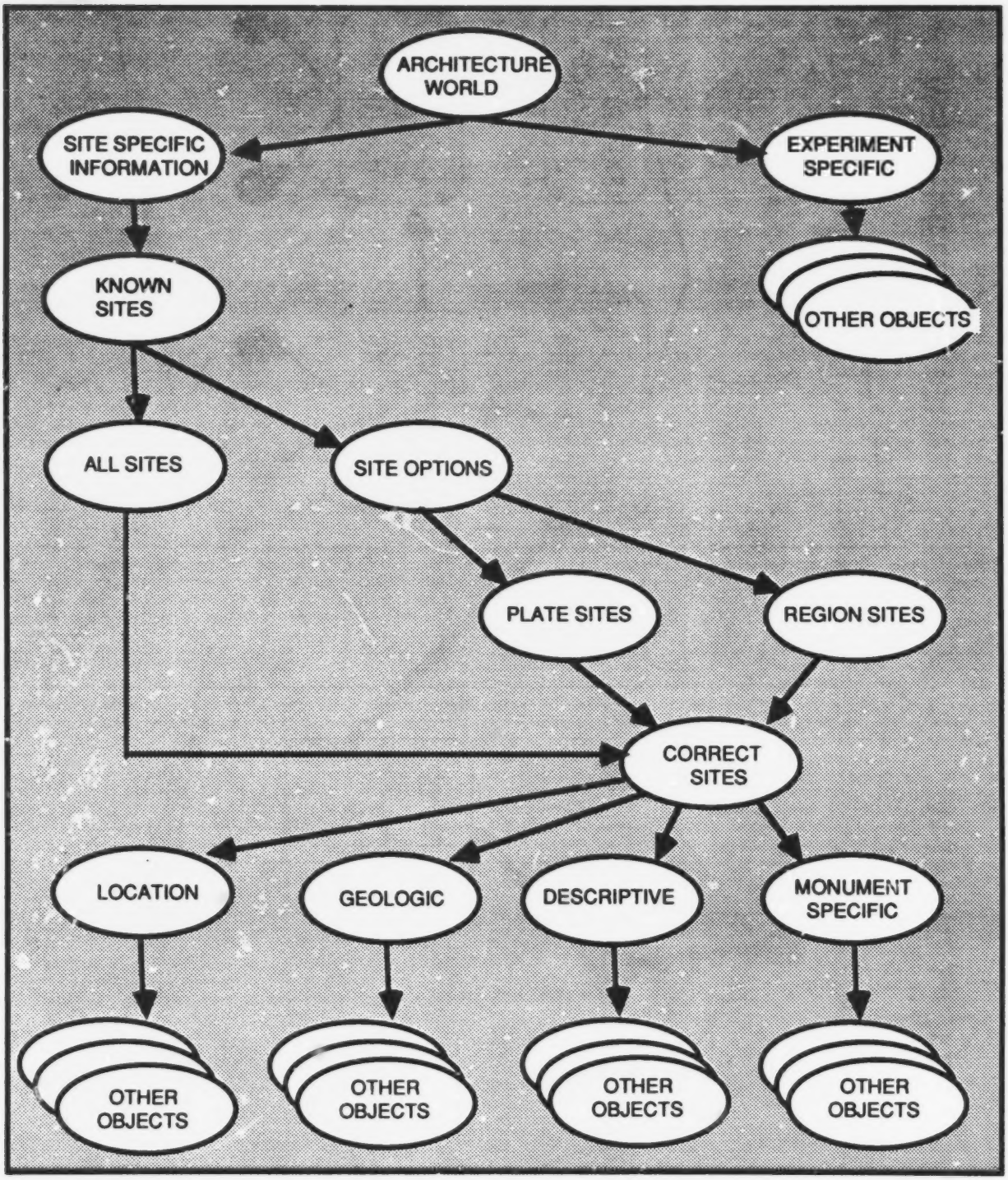


Figure 13. Architecture View

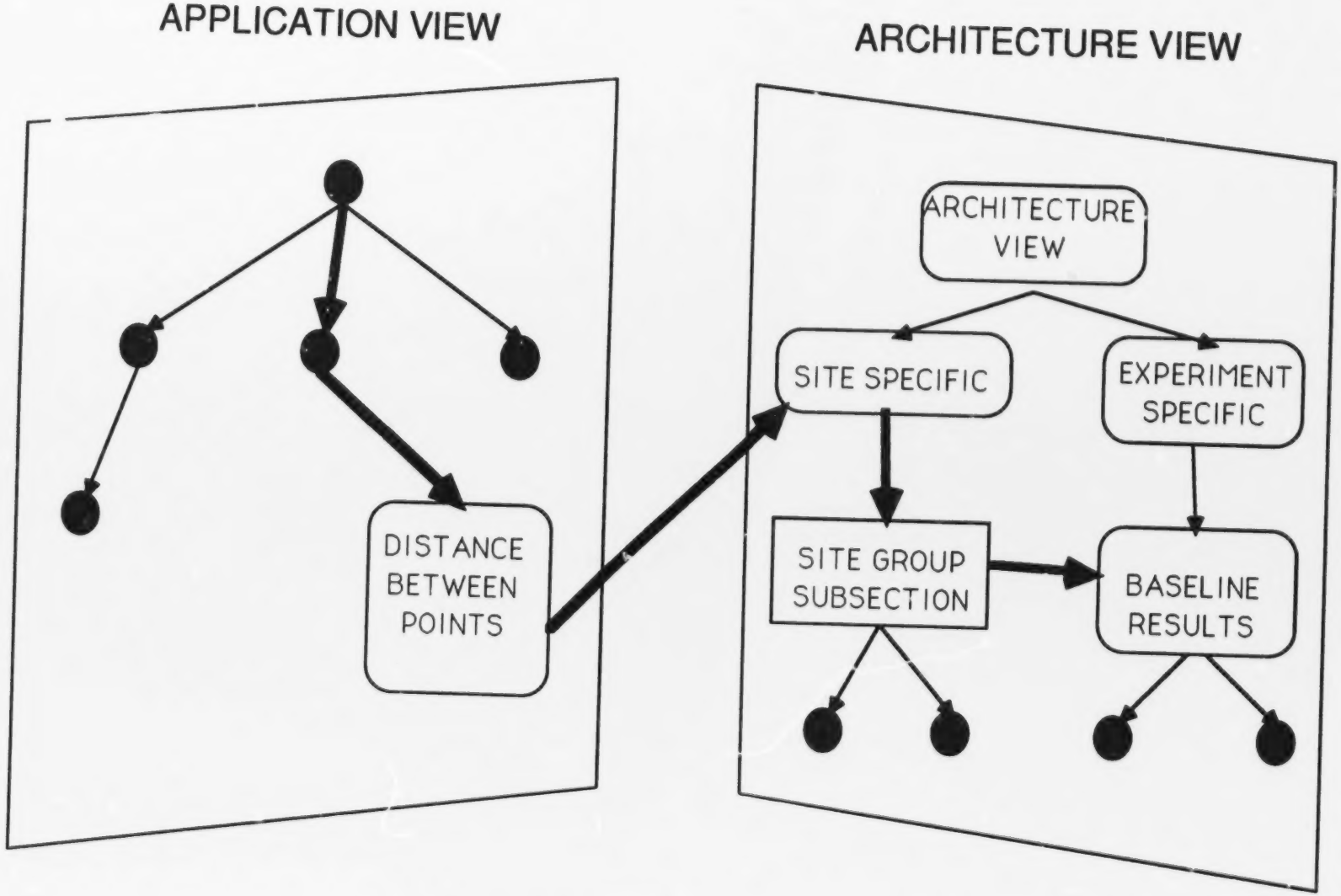


Figure 14. Distance Between Points

word or phrase that is associated with a collection of fields. In the sample display,

"You have two equivalent queries for getting the information. Either you can:

a) show me the surroundings, or
b) show me the current name, location, plate and site description,"

the user has the option of either sending a) or modifying b) (i.e., project, select, etc.) by not requesting a field or set of tuples. This option, in fact, reflects a major goal to instruct the user because only the phrase "the surroundings" needs to be recalled as a class descriptor.

This mnemonic for the fields also forces a performance issue. Because databases contain large sets of objects, an expert system must integrate them into the KB in order to reason about them. By storing only mnemonics such as "surroundings" in the KB, the field combinations can be stored externally and loaded as needed. This technique serves not only to compact data but also to aid maintenance, inasmuch as modification of fixed format files is easier than changing code.

Keeping the instruction goal in mind, the third sub-module allows the user to request definitions of unclear terminology used by the system. The technique used here is similar to the key-word pattern matcher in the transform filter [1] except that a dictionary must be loaded.

The second major component of the Crustal IUI is the customized natural language front-end, THEMIS (see Figure 15 for a description of the functional components of the system).

Once an English query is entered from either CRUDDES or the communications package, the THEMIS driver passes it to the THEMIS server, the database-customized section, which translates the English query into an interlingua called TQL. After receiving the TQL equivalent, the driver translates it into a SQL equivalent for ORACLE. Finally, the relational information is retrieved by ORACLE and passed back to the driver, which returns the information to the user over the communications line.

Prior to customization, the THEMIS system contains an ATN-like (Augmented Transition Network) parser and a lexicon of common English words and parts of speech. Functionally, only the driver is a fixed part of THEMIS, whereas the server image must be created to contain the customized sections.[7]

Much of the customized lexicon, that is generated by the learning statements called for by THEMIS, actually corresponds to a completion of all possible terminal nodes on an ATN parse tree and a linkage to the database objects. For example, suppose the following simple query

"Show site name, plate, and station"

were entered. After the application of an imperative case transformational rule[5] to the initial parse, the parse tree in Figure 16 could result from the grammar rules shown in Figure 17. Again, the circled rule "V —> show" and "N —> you " existed in the common English lexicon, whereas the squared rules were results of learning statements. In fact, the lexicon definition file contains statements like

(actual learning statement)
TEMPERATURE means field TEMP
SWNOUN T PERMDEF T

that mean TEMPERATURE is a single word, is also a noun, and is a permanent definition in the definition file.

To customize the server, two processes are activated. First, the definitions or learning statements are entered in interactively in one of two modes, namely, global or temporary definition mode. In the global mode, every definition affects all users, whereas in the temporary mode, the effect is confined to the specific end-user making the definition. This allows for the creation of personalized lexicons which reflect the user's terminology and perspective. Second, when appropriate intervals of time, the server image must be stored using the command "makeremoteimage."

CUSTOMIZATION LIFE CYCLE

Because much of customization involves establishing the parser's terminal nodes, development of the applications-specific interface can often be done using the following steps.

Restructure Database By Using Oracle Views

Design the THEMIS system with the help of the database administrator before creating database views. These special THEMIS views are created because of the duplication of names in fields, tables, and THEMIS reserved words. Duplication of names cause problems because THEMIS considers any fields with the same name to be join fields (i.e., foreign key fields mapped to the primary key). In other

PRECEDING PAGE BLANK NOT FILMED

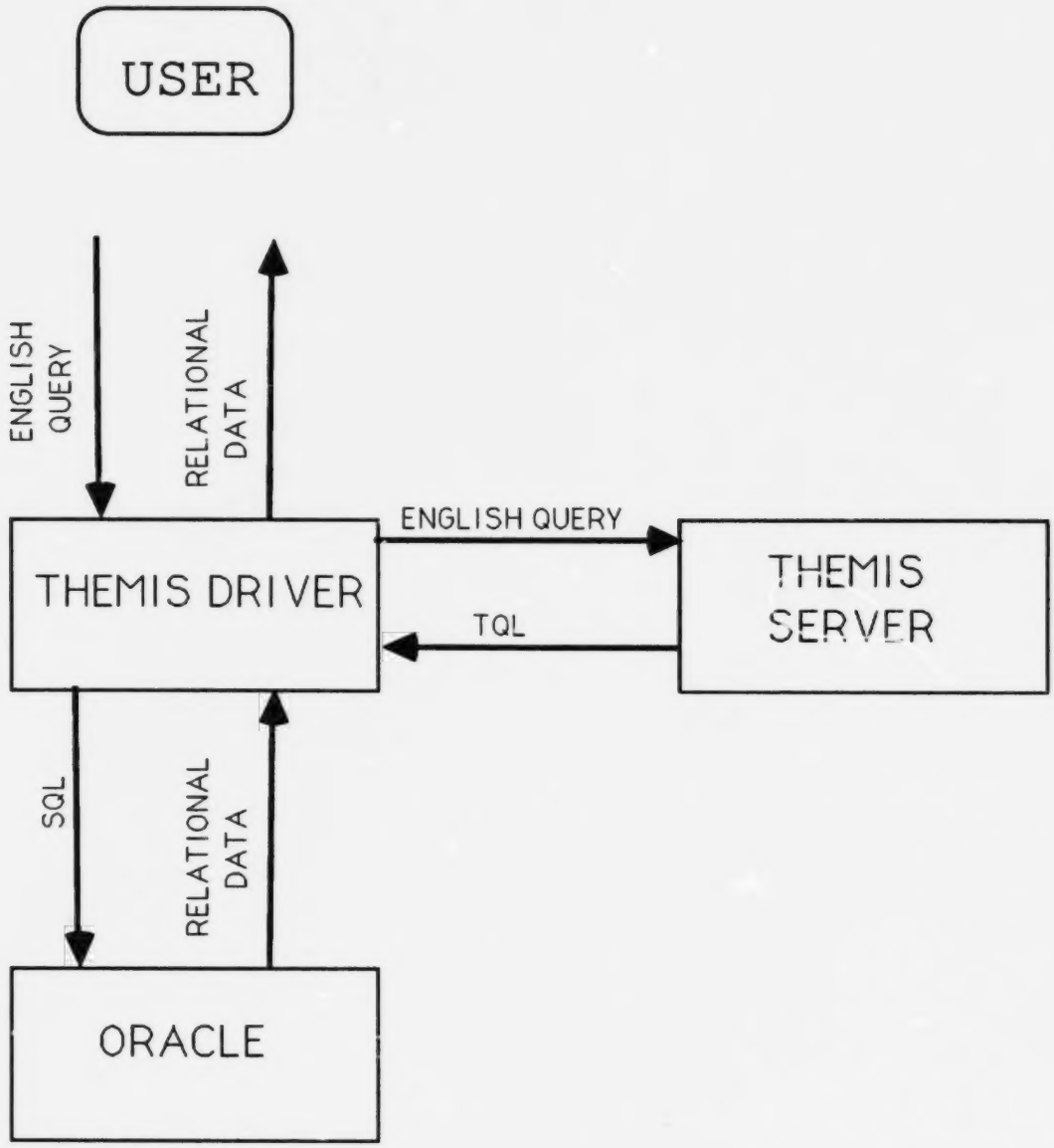


Figure 15. Themis Systems Components

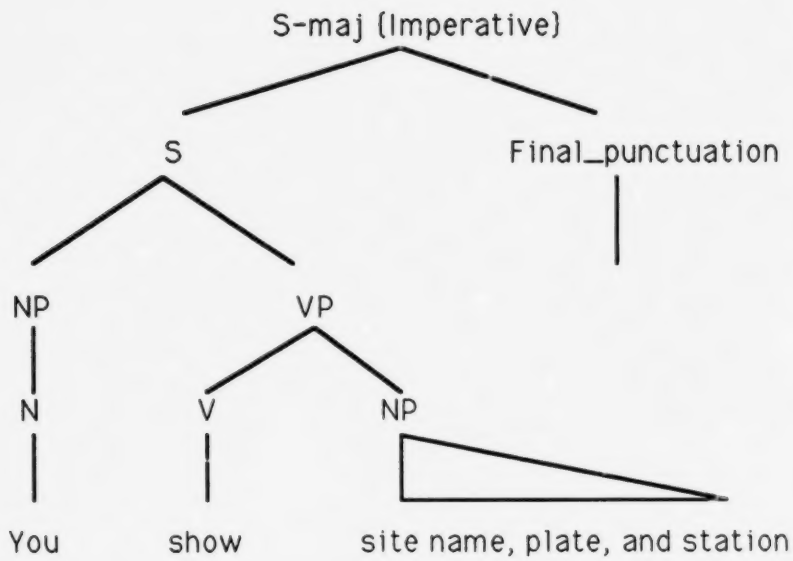


Figure 16. ATN Parse Tree

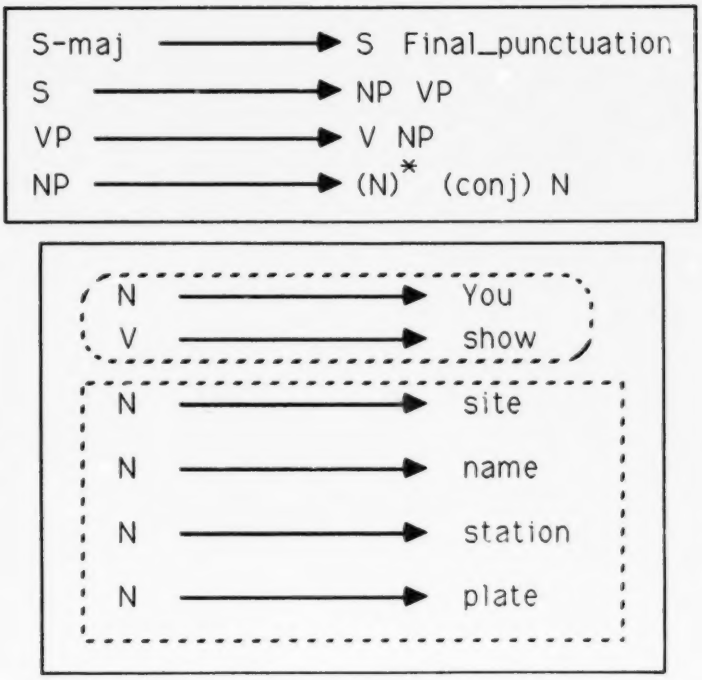


Figure 17. ATN Subsection Grammar

words, a join occurs when the request data can be satisfied from more than one table in the database and two or more fields from these tables contain identical field names. If any names are identical where a legal or undesired join can happen, an illogical result could occur. For example, the class of baseline tables (Baseline78_slrgsfc, Baseline79_slrgsfc, ...Baseline83_slrgsfc) each have the same attribute names (F_STATION, S_STATION, BASELINE,.., GEODESIC_SIGMA). Therefore, any query to a baseline table, whether or not the time or experimental method is specified, will always create a set of joins between all the tables of this class.

Unfortunately, this example presents another problem when dealing with classes of similar tables and fields. Since tables are sequentially linked to the database using a special learning statement (ACCESS table..etc.), only the last reference to the table of the class is remembered by THEMIS because of the similarity in field names. Thus, only Baseline83_slrgsfc would be recalled by THEMIS in the above example.

Acquire Database-specific Information

Assemble all information about the database: table names, field names, field data types, table sizes, indices (fields that expedite database access), and join fields.

Classify Database Information

Sort out classes of the fields and tables so as to eliminate duplicate field names, unless the join of the fields is considered permissible. Also, no identical table names should exist.

Changes Duplicates

Modify the appropriate field names and/or table names if they are found to be duplicates. Because of the cumbersome task of "unjoining" unwanted fields, caution should be taken to ensure that all changes are made prior to further customization. To avoid duplicate field or table names which contain unrelated information, it is advisable to consult with the database administrator, applications database experts, and potential users.

Determine Synonyms

Interview the database development experts and potential users to determine their terminology and concepts regarding the database. Then, create a list of synonym(s) for all the fields in tables of the database.

Teach THEMIS

Enter the information incrementally and interactively. Note that THEMIS asks questions when needed to resolve word usage.

Test THEMIS

Test as many entries against the database as possible, comparing the information retrieved by THEMIS with the information retrieved by ORACLE using SQL. Using the tools available in THEMIS, show the final THEMIS SQL query and compare the THEMIS generated query to the actual SQL query.

Compress New Statements And Makeup-Dated Images

Compress the THEMIS memory tables prior to making an executable remote image to reorder THEMIS' storage of the tables and fields. Lastly, to ensure proper archiving, set up an executable remote image file of the entries that yield satisfactory results periodically.

Exit THEMIS

Exit THEMIS with a "BYE." command as this will update the definition file containing the tables and their field names.

Repeat Steps 6 Through 8

Save Remote Image

LEARNING STATEMENTS

The following are the four basic categories of learning statements in THEMIS:

- INITIALIZATION statements initialize the THEMIS system by mapping it to the ORACLE domain. These include accessing the tables, renaming the appropriate table or field names, and identifying the join or index fields.
- KNOWLEDGE statements provide the synonyms and class descriptions for linking to the Crustal database objects.
- PROSE statements add to the design and decor of the output from the queries.
- EFFICIENCY statements assist in faster data access (e.g., knowing the number of records in a table or the location of table indices).

Appendix B. contains the list of the generic learning statements and their definitions, Figure 18 shows an example learning statement for the DIS, and Figure 19 shows a sample THEMIS session.

LIMITATIONS OF NATURAL LANGUAGE QUERY PROCESSOR FRONT-ENDS

Word-sense and *referential* ambiguity comprise the major limitations of natural language query processors as well as compositional natural languages in general.[5] The first involves a problem with word meaning whereas the second arises when an entity like pronouns have no clear (anaphoric) reference to any regular noun.[12] Section 7 previously addressed the problem of word sense ambiguity. The problem of referents in the database world occurs in the join-net, as seen in Figure 20. In order to resolve both kinds of ambiguity the context of the situation, determined in the ES, must specify a query before the NLQP (Natural Language Query Processor) receives it.

This issue of context and relativism cannot be solved in THEMIS, in particular, because of its lack of ability to deal with two types of errors: intensional and extensional failure. These errors are distinguished by designating *intensional* failure (1) as an error a user makes about his belief of the database structure and *extensional* failure (2) as an error the system makes about assigning an interpretation (i.e., the correct data) to the user's request.[11]

As an example of 1 (intensional), suppose a user requested

> "What are the satellites used in the VLBI method?"

DATA DICTIONARY

SITE_INFO

COLUMN NAME	DESCRIPTION
I_DATE	Data insertion date (DD-MON-YY)

THEMIS LEARNING STATEMENTS

LEARNING STATEMENT	DESCRIPTION
ACCESS TABLE site_info USING PATH NULL.	Connects site_info to ORACLE tables
Data Insertion Date IS HEADING FOR FIELD i_date	Formats output for headings
i_date IN site_info IS AN EVENT.	Makes i_date into a date field for searching and output formatting

Figure 18. Example Learning Statements

```
THEMIS
Welcome to THEMIS Management Information System
by FREY ASSOCIATES, INC.
Artificial Intelligence Division
TM THEMIS is a trademark of FREY ASSOCIATES, INC.
THEMIS server version 1.1.08.BAC 07-May-1985
THEMIS driver version 1.1.07.SQL 10-May-1985

I am designed to answer queries about the contents of your database. To ask a ques-
tion, just enter it in ordinary English, terminating your question with a period, question
mark or exclamation point.

To find out about the kinds of information in your database, just ask me:  What do you
know about?

Enter your response:
What do you know about?

The database available to me contains tables about the following items:

STARLETTE_85
STARLETTE_84
STARLETTE_83
EXPERIMENTS_VLBIGSFC
BASELINE_VLBIGSFC
ANCILLARY_LLRTEXAS
STARLETTE_81
STARLETTE_80
STARLETTE_79
STARLETTE_76
SITES_SPECIFIC
SITE_INFO
OCCUPANCY_VLBI
OCCUPANCY_SLR
NORMALPT_LLRTEXAS
LAGEOS_85
LAGEOS_84
LAGEOS_83
LAGEOS_82
LAGEOS_81
LAGEOS_80
LAGEOS_79
LAGEOS_78
LAGEOS_77
LAGEOS_76
```

Figure 19. Example Themis Session

```
EXPERIMENTS_VLBIJPL
COORDINATES_SLR
CALIBRATION_SLR
BEC_85
BEC_84
BEC_83
BEC_82
BEC_81
BEC_80
BEC_79
BEC_78
BEC_77
BEC_76
BASELINE_VLBIJPL
BL83_SLRGSFC
BL82_SLRGSFC
BL81_SLRGSFC
BL80_SLRGSFC
BL79_SLRGSFC
BL78_SLRGSFC
For additional information regarding these tables, ask
what do you know about table TABLENAME?

Substitute one of the above table names for the word TABLENAME.
For example:
What do you know about bl78_slrgsfc?

Enter your response:
Show me all the plates.
PLATE
_____
NORTH AMERICAN
PACIFIC
CARIBBEAN
SOUTH AMERICAN
EURASIAN
AFRICAN
NAZCA
AUSTRALIAN-INDIAN
ARABIAN

Enter your response:
Bye.
Thank you very much for spending this time with me.
See you soon!
```

Figure 19. Example Themis Session (Continued)

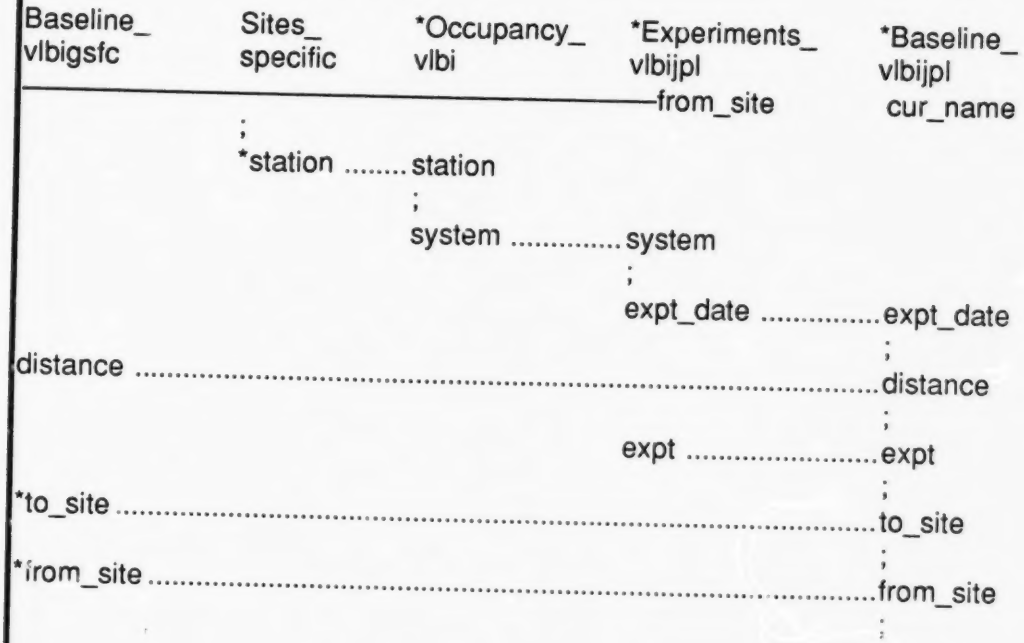
```
Enter your response:
> THEMIS show driver query.

Enter your response:
> Show distance in baseline_vlbigsfc in 78 for the north american plate
> where from_site in basline_vlbigsfc equals cur_name in sites_specific.

Select baseline_vlbijpl.distance, baseline_vlbigsfc.obs_date, plate from baseline_vlbigsfc,
occupancy_vlbi, sites, experiments_vlbijpl, baseline_vlbijpl where (baseline_vlbigsfc.obs_date
>= '1-JAN-78` and baseline_vlbigsfc.obs_date < '1-JAN-79` and sites.plate = 'NORTH AMERI-
CAN' and baseline_vlbigsfc.from_site = sites.cur_name and occupancy_vlbi.station and
experiments_vlbijpl.system = occupancy_vlbi.system and baseline_vlbijpl.expt_date =
experiments_vlbijpl.expt_date and baseline_vlbigsfc.distance = baseline_vlbijpl.distance and
baseline_vlbijpl.expt =experiments_vlbijpl.expt and baseline_vlbigsfc.to_site =
baseline_vlbijpl.to_site and baseline_vlbigsfc.from_site = baseline_vlbijpl.from_site).

Do you want to proceed with the query? y or n [y]
n
```

Notice that the joins were made at:



Note:
* - These tables and/or their fields ought not to have been involved!

Figure 20. Join NET Problem

THEMIS would then respond "no records selected."

Given this response, the user may infer either:

(a) no satellite types have yet to be used with VLBI or,
(b) no satellites can be used with VLBI.

The proper response to this "non-existent relationship" problem might involve (b), as in

"satellites use SLR" and
"Quasars use VLBI."

As implied above, an extensional error occurs when, despite a correct parse, the internal semantic representation mismatches the user's intent. While this may occur because of the associated problems with word sense and referential ambiguity, much of the problem is directly attributed to a superficial customization. Regrettably though, further customization only increases the probability of correct interpretation (i.e., never guarantees correct interpretation), as rarely, in practice, can a NL interface be inclusive to a language.[12]

LIMITATIONS OF THEMIS IN THE DEVELOPMENT ENVIRONMENT

THEMIS, as with most ATN-based systems, requires extensive use of backtracking from incorrect parses, large dictionaries with multiple parts of speech per word, and many transformational rules. Such criteria, unfortunately, often require significant computational power. The Crustal natural language front-end, in fact, typically requires a non-trivial amount of the CPU's utilization (VAX 11 /780) and 21 megabytes of disk storage. Such demanding resource requirements dictate the need for a dedicated machine, by today's standards.

GRAPHICS INTERFACE

A major drawback in the IUI system is its inability to communicate in a format different from verbal questioning or the English language. Since geographically based systems by nature tend to be spatially oriented, a method is needed to select information from a spatial or graphics context. In fact, several places exist in the expert system where information, such as site location, has meaning in a global context. Continuity of the problem domain can be important in assisting problem solving. For example, in the applications view, graphically presenting the continents superimposed on the plates is more natural than listing them in a question such as:

Choose two plates from this list:

p) Pacific	n) Nasca
na) North American	c) Carribean**
sa) South American	e) Eurasian
a) Australian	af) African**

**no data yet from these plates.

With these needs in mind, a separately funded project was initiated to produce a graphics interface to the CRUDDES environment. In general, by using a mouse, the interface allows the user to choose spatial objects such as plates, sites, and baselines from several map projections of the Earth onto the two dimensional plane of the CRT display (e.g., mercator, ortho-graphic, etc.). Furthermore, ancillary functions, such as an instruction line, drop down menus for help, display of the legend, etc. can be added to guide the spatial selection process.

OVERALL GRAPHICS INTERFACE DESIGN

Because of the intention to create a prototype, only two scenarios that utilized the graphics are currently integrated into the expert system. Whereas the first scenario encompasses the aforementioned problem with the plates, the second scenario involves choosing information about sites in the architectural view.

As previously mentioned in the section on the architecture view (see Figure 12), the list of sites must first be diminished to ease retrieval of site ancillary information. As it turns out, the need for this information may not be set in the context of the application view, as this view deals only with baseline objects. For one thing, expert database users would be interested in ancillary information about sites for a specific region (e.g., California fracture zone), set of regions, or the globe. Therefore, setting the context for the pruning of the sites translates into establishing the geographical boundaries of interest.

These criteria are manifested in the graphics interface by first showing a projection of the entire world with (VLBI and SLR) site locations marked and then using the mouse to "view port" or "zoom in" over a region of interest. In this manner, the user can go from the global level to the site level choosing sites along the way. The result, therefore, is a smaller list of sites that is returned back to the expert system. The reduced site list acts as a qualifier in the search for additional site-related information and greatly reduces computertime and storage requirements in the search.

To implement the graphics interface, a GKS (Graphics Kernel System) derivative, called the Graphics Software Systems (GSS), is used for rapid development, language compatibility, and device independence. Specifically, the interface was functionally developed separately in the C language. Linked to the expert system using the M.1 external interface, these modules in C contain calls to the GKS package for actual display rendering.

Syntactically, external functions are called

from within the rule's antecedents in the KB. For example, the rule

rule-a12:

IF 'plate stabilization' and
external(graph,[1,1]) = [REF] and
THEN select-moving-plate = REF.

calls the function "graph" with the input and output parameters "[1,1]" and "[REF]" respectively.

DETAILED GRAPHICS DESIGN

Figure 21 summarizes the complete system on the PC/AT, including graphics. In addition to the aforementioned graphics modules, several other modules of interest exist. For example, the foreground capture module ingests a subset of the DIS into the PC/AT so as to provide current information about sites such as the site number,latitude/longitude, location name, site type, etc. Specifically, this information is used to project and graphically represent the location of the site along with the text data.

Despite the intention to improve system performance by localizing this data, a problem with the data integrity can arise if this process is not automatic. For example, if more sites are added to the DIS and the foreground capture is done manually, sites will be missing from the global map display after the only foreground capture is done. To ensure site integrity, therefore, the file capture program exists and can be called to automatically recapture the local foreground data in a form amenable to GKS.

Like this capture mode, the background capture module ingests data from another data set source on the VAX 11/780. The difference, however, is that the ingested data consist of digitized images and world coastal data stored in latitude/longitude coordinates. In essence, these background objects set the context for the foreground objects to be plotted.

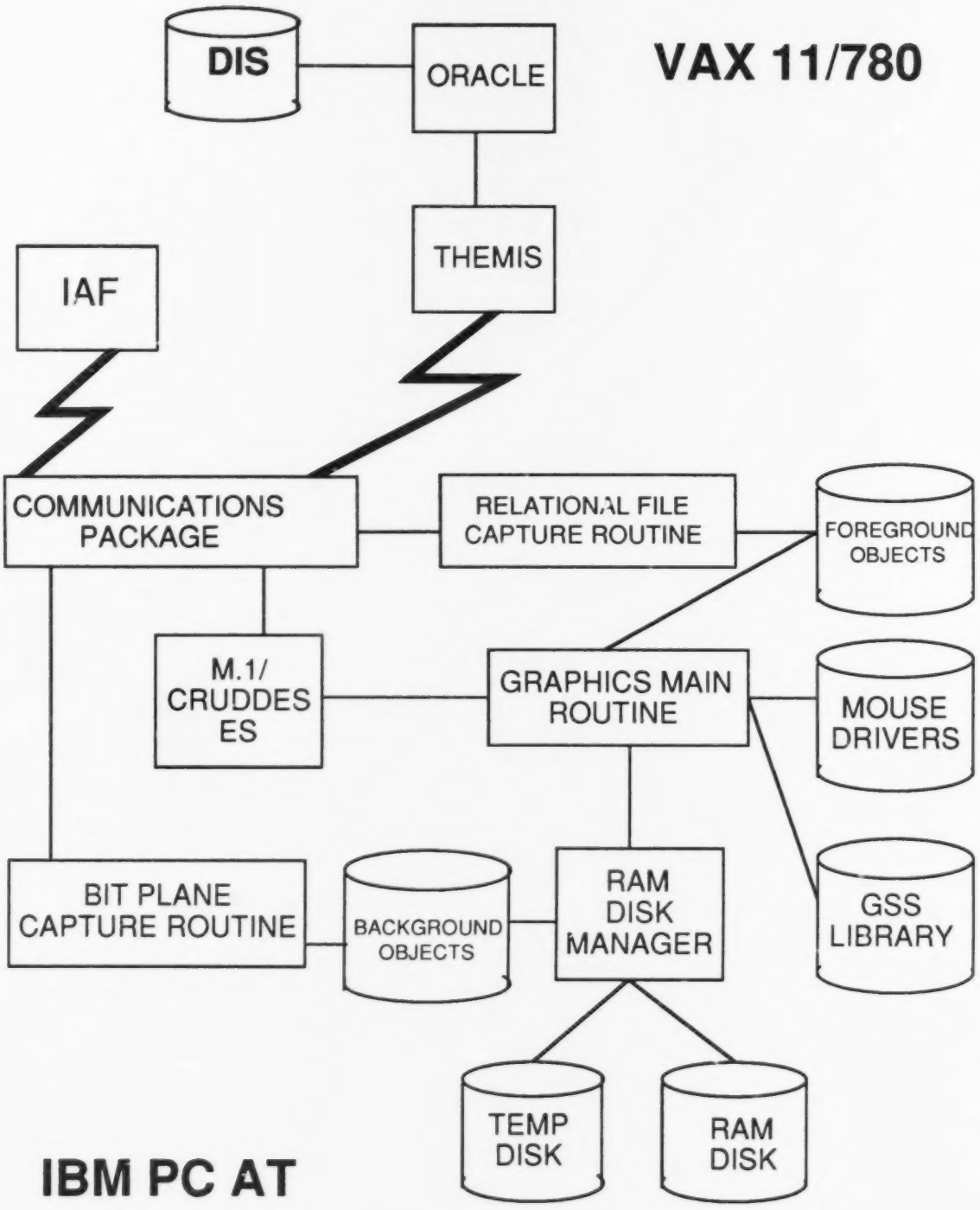


Figure 21. System Design

PROTOTYPE CONCLUSIONS AND FUTURE DIRECTIONS

The establishment of context permeates the entire design and is, in fact, the major contribution to the database domain. As was seen with natural language front-ends, context is essential in resolving ambiguity problems with word sense and referents. Likewise, graphics demands spatial context in order to clarify the set of foreground objects. This spatial clarification amounts to a circumscription of the foreground objects by the use of map borders and transformation functions which map geographic coordinates to a raster display and back.

A generic or initial context in the database can, therefore, be defined as a view of the database from the knowledge or graphics base. Each view, whether of an applications, architectural, or graphical type, contains different types of knowledge. Of course, the amount of knowledge-type uniquely determines the characteristics or type of view. For example, the applications view uses more heuristic knowledge while the graphics depends heavily on procedural knowledge. Therefore, a major development issue involves determining how to exploit these characteristics within each view type using the current tools.

Hence, a major result of the prototype project involved identifying the following technical limitations in the tools.

1) [NLQP] As more knowledge and ability for clarifying ambiguation is imbedded in THEMIS' domain, system performance degraded non-linearly.
2) [M.1] Because, by their nature, the applications and architecture views are goal-driven (backward chaining) and data-driven (forward chaining) respectively, M.1, which currently supports only backward chaining, cannot handle the architecture view in its appropriate data-driven manner.
3) [M.1] Since much of the IUI deals with real and virtual objects, M.1 cannot deal with objects and their features in a manner other than describing them as strings or tokens (as in section 7.2).
4) [M.1] Because each rule globally inherits all conclusions from previous rules in the cache, no automatic method (i.e., default logic) for controlling the cache exists other than a manual manipulation using meta-facts or meta-propositions.
5) [M.1] Without a natural mechanism for representing views of the database, views must be represented as rule modules that are placed in a specific order.
6) [M.1] Without a natural method for dynamically generating graphics within the M.1 environment, duplication of information may occur between the knowledge base and graphics, causing program integrity problems.
7) [GSS] Because no hierarchical data structures exist for creating and manipulating composite graphical objects (e.g., for different levels of resolution), much of the graphics development amounts to creating these structures from a bottom-up or detailed level.
8) [PC/AT] Since the PC/AT uses a "segment:address" architecture, non-standard "C" data types have to be used to create bit-map images larger than 64k.
9) [DOS/M.1] Because of the 640k limitation of IBM's Disk Operating System Ver. 3.0 and the lack of a robust garbage collection facility in M.1, dynamic memory frequently becomes exhausted, creating an unstable operational environment.

PRECEDING PAGE BLANK NOT FILMED

The solution to the first problem involves minimizing knowledge use in the NLQP. The latter problems associated with M.1, as discussed below, demand the use of a more robust development tool, such as the Automated Reasoning Tool (ART) (version 3.0, by Inference, Corp) as discussed below.[9]

In addition to handling both backward and forward chaining (Limitation 2), these advanced tools (e.g., ART) organize data into a schema-based, inheritance architecture. This paradigm, in fact, simplifies the task of describing virtual objects as well as controlling the conclusions of previous rules (Limitations 3-4) and a natural mechanism for creating both pre-existing and hypothetical viewpoints (Limitation 5). While the former viewpoint allows for the pre-existing classes of users, or the applications views, the latter enables the system to maintain personalized views for each user. In other words, the system hypothesizes about the user based on his previous responses or consultations in order to arrive at a personalized view or model. Moreover, these tools contain a graphics interface (Limitation 6), that allows graphics icon generation and bit-maps within the same environment. Another advantage of the advanced graphics interface is that graphics objects are represented as schema hierarchies (Limitation 7). Having the ability to reference these objects amounts to an instantiation of the object's icon name into the objects schema or frame. The icon can, therefore, be thought of as an attribute of a database object. Lastly, implementation of these tools on a LISP machine solves problems of Limitations 8 and 9.

Since these tools are already "in-house," the goals of the Intelligent Data Management project over the next three to five years are to develop the following:[1]

- A sophisticated system interface using advanced tools,
- An intelligent automatic data ingest and maintenance system,
- A spatial database system linked to an IUI, and
- A dynamic database system that will perform complex data management operations "on-the-fly."

Concurrent with the above technical development effort, the project will select a suitable candidate test-bed project, such as a Science and Applications Information System (SAIS) or Earth Observing System (EOS), to use for evaluation. Once selection has been made, a test-bed system will be customized based on the data management needs and technical direction of the selected project.

APPENDIX A META-FACTS AND META-PROPOSITIONS.

The following are some of the meta-facts along with their descriptions, used in CRUDDES[8]:

"nocache(EXPRESSION) :

Values of nocache expressions are not stored permanently in the cache. When such expressions are evaluated, the results are temporarily cached and the proposition that depends on the results is tested. Then, before proceeding, M.1 deletes the facts from the cache. If the value of a nocache expression is needed later in a consultation or session, M.1 seeks and re-creates it again.

whenfound(EXPRESSION) = LIST or
whenfound(EXPRESSION = VALUE) = LIST :
A whenfound knowledge base entry allows the modification of the M.1 inference process by designating what M.1 should do whenever a value for the EXPRESSION is found.

noautomaticquestion(EXPRESSION) :

This overrides M.1's default behavior of generating a question to the user whenever an expression remains unresolved, even if there are no other knowledge base entries for the expression. If the expression is a variable, then it turns off the automatic question facility for all expressions.

explanation(LABEL) = TEXT :

Explanation displays the string TEXT in response to "why" queries regarding a knowledge base entry whose label matches LABEL. The text must be either an atom (i.e. smallest divisible object--character or word) or a list whose elements are atoms and formatting commands("nl" and "tab"). This meta-fact enables knowledge engineers to substitute customized explanations or to offer acceptable explanations about a path of reasoning that contains proprietary information. LABEL can be any expression and may contain variables. Ordinarily, rules, presuppositions, whenfounds, and initial data are the entries used with the"explanation" meta-fact.

legalvals(EXPRESSION) = integer(LOW, HIGH) :

An integer between LOW and HIGH inclusive is a suitable response as a value for this expression.

legalvals(EXPRESSION) = LIST :

The elements of the list are acceptable values for the expression. M.1 attempts "auto completion" (meaning that the answer must be one of a set of fixed responses and only sufficient letters to unambiguously distinguish the response need be typed or M.1 will repeat the question) on responses to identify an element from this list.

legalvals(EXPRESSION) = number :

Any number is an acceptable value for EXPRESSION.

legalvals(EXPRESSION) = number(LOW, HIGH) :

The real or integer numbers given as responses must lie within the range LOW to HIGH, inclusive.

legalvals(EXPRESSION) = real :

Any real (floating point) number is a suitable response. A real number must include a decimal point with at least one digit on either side, unless an exponent is used. Either fixed or floating point notation are acceptable.

legalvals(EXPRESSION) = real(LOW, HIGH) :

The real number given as a response must lie within the range of real numbers, LOW to HIGH, inclusive."

APPENDIX B THEMIS GENERIC LEARNING STATEMENTS.

The generic types of learning statements recommended in Crustal IUI are:
(Note: The uppercase letters denote THEMIS key words and the lower case letters, user-supplied.)

a. ACCESS TABLE tablename USING PATH oracle_userid.

   When THEMIS accesses a table from ORACLE, it reads the names of the table and columns, the column lengths, and types, after which, the table name and column names become part of THEMIS' vocabulary. THEMIS echoes the column names on the screen as it is accessing the table, and when THEMIS is exited, it updates the information about the table in the dictionary file, THEMIS.DIC.

b. i. WHEN YOU SHOW field1 ALWAYS SHOW field [field...].
   This statement is used to display different fields that go together when field 1 is requested by the user.
   ii. WHEN YOU SHOW INFORMATION ABOUT table, ALWAYS SHOW field [field...].
   This statement is used to display the field(s) that is to be displayed when a table is requested.

c. i. category IS A CATEGORY IN table.
   This statement declares a category or "group" of fields.
   ii. category CONTAINS field1 [,field2...].
   This statement declares the fields that belong to that category.

d. i. field IS THE DEFAULT CHARACTER FIELD FOR table.
   When a request contains a word that THEMIS does not know, THEMIS tries to see whether it is a data value in the requested table by matching the value to values in the table's default character field.
   ii. BY DEFAULT, DISPLAY field [,field...] FOR TABLE table.
   This statement is used by THEMIS to display data whenever it cannot identify the particular fields to display from the table the request is referencing.

e. DISPLAY number DIGITS FOR FIELD field.

This statement is used to instruct THEMIS as to how many digits/characters are to be displayed for numeric or calculated fields.

f. i. datefield IN table IS AN EVENT.
ii. field IS A 4 DIGIT YEAR.
iii. field IS A 2 DIGIT YEAR.
These statements assist THEMIS in queries regarding dates. The only difference between them is that the datefield in (i) has to be of type date and the fields in (ii) and (iii) can be numeric.

g. A number TRUNC IS THE DEFAULT FORMAT SPECIFIER FOR FIELD field.

This statement is similar to (e), except that it is used for character and date fields, for example, A12.

h. heading IS THE HEADING FOR FIELD fieldname.

This statement both defines the title for a field and also describes the field.

i. field IS AN IDENTIFIER.

This statement makes a numeric field that does not represent a quantity into a numeric code. This enables THEMIS to count them and also reminds THEMIS that it may not perform any arithmetic operations on that field.

j. i. FIELD field IS UNIQUE [IN table].
ii. field AND PLURALS ARE UNIQUE [IN table].
These statements enable THEMIS to import the field(s) by using the field values in requests. The field values have to be unique.

k. i. ADD field [IN table] TO THE VOCABULARY.
ii. ADD field [IN table] AND THEIR PLURALS TO THE VOCABULARY.
These statements perform the same function as those in (j) except that the field values are non-unique.

l. field IS INDEXED [IN table].

This identifies which Fields in the database are indexed by the DBMS.

m. FIELD fieldname1 IN tablename1 MAPS INTO FIELD fieldname2 IN tablename2.

This statement tells THEMIS to treat these two fields with different names but common values as join fields.

n. field1 IN table1 IS NOT EQUIVALENT TO field2 IN table2.

This statement tells THEMIS to not treat these two fields with same names but different values as join fields.

o. name IS THE PRIMARY NAME FOR tablename.

This statement defines an English-like name for each table in the database.

p. reportname MEANS PRINT THAT.

This statement declares a report name to THEMIS to generate printed reports.

q. i. ASSUME TABLES CONTAIN number RECORDS.
ii. TABLE table CONTAINS number RECORDS.
These statements are similar. (i) specifies an average size of all the tables and is to be used if that is so, else use (ii) where each table size is declared individually.

r. number DIFFERENT VALUES FOR fieldname.

This statement tells THEMIS how many values there are to a particular indexed field.

s. i. USE THE NAME name FOR TABLE tablename.
ii. USE THE NAME name FOR FIELD fieldname.
These statements are used to rename a table or field.

t. field RESIDES IN table.This statement tells THEMIS that a particular join field with unique values is assigned to a particular table.

u. i. singular IS THE SINGULAR OF plural.
ii. plural IS THE PLURAL OF singular.
These statements define the singular or plural form of a noun.

v. i. synonym MEANS FIELD fieldname.
ii. synonym MEANS TABLE tablename.
iii. synonym MEANS phrase.
iv. synonym MEANS field = "value" [,field = "value"...].
v. synonym MEANS value [,value...].
A synonym is an English phrase that is composed of nouns, verbs and adjectives that represents a field, table or another synonym. A synonym can also be used to represent one particular value of a field, a group of field values, or even imported field value(s).

w. phrase MEANS calculation.

This statement assigns an arithmetic calculation to a phrase or synonym.

x. i. verb IS A VERB REFERRING TO FIELD field.
This statement tells THEMIS that a verb refers to a field so that the verb can be used in requests about the field.
ii. THE VERB verb TESTS FOR FIELD field.
This statement defines a presence verb for fields that may not have a value in each record. When used in requests, THEMIS checks to see if the record has any value and only selects those records that do.

BIBLIOGRAPHY

[1] Campbell, W.J., Roelofs, L.H., and Short, N.M. Jr., "The Development of a Prototype Intelligent User Interface System for NASA's Scientific Database Systems," NASA TM 87821, April, 1987.

[2] "Global Geodynamics," Geodynamics Program Office, NASA Headquarters.

[3] Date, C.J., An Introduction to Database Systems, Vol. I, Forth Edition, Addison-Wesley Publishing, Co., copyright 1986.

[4] Sowa, J.F., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing Co., copyright 1984.

[5] Charniak, E., and McDermott, D., Introduction to Artificial Intelligence.

[6] Barr, A., and Feigenbaum, E., The Handbook of Artificial Intelligence, Vol. I, William Kaufmann, Inc., copyright 1981.

[7] THEMIS User's Guide, Frey Associates, Inc., copyright 1985.

[8] M.1 Reference Manual, Teknowledge, Inc., copyright 1985.

[9] Clayton, B.D., ART Programming Tutorial, Vol. I-III, Inference Corp., copyright 1986.

[10] Bic, L. and Gilbert, J., "Learning from AI: New Trends in Database Technology," IEEE Computer, March 1986, P. 44-54.

[11] Mays, E., "Failures in Natural Language Systems: Applications to Database Query Systems," Proc. First Natl. Conference on Artificial Intelligence, 1980, P. 327-330.

[12] Rich, E., "Natural-Language Interfaces," IEEE Computer, Sept. 1984, P. 39-47.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM 100693		

4. Title and Subtitle	5. Report Date
The Crustal Dynamics Intelligent User Interface Anthology	October, 1987
	6. Performing Organization Code: 634.0

7. Author(s)	8. Performing Organization Report No.
Nicholas M. Short, Jr., William J. Campbell, Larry H. Roelofs, and Scott L. Wattawa	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
Goddard Space Flight Center Greenbelt, MD 20771	13. Type of Report and Period Covered: Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546-0001	

15. Supplementary Notes

Nicholas M. Short, Jr., National Space Science Data Center, Data Management Systems Facility, Goddard Space Flight Center; William J. Campbell, National Space Science Data Center, Data Management Systems Facility, Goddard Space Flight Center; Larry H. Roelofs, Computer Technology Associates, McLean, Virginia 22102; Scott L. Wattawa, Science Application Research, Inc., Lanham, Maryland 20706.

16. Abstract

The National Space Science Data Center (NSSDC) has initiated an Intelligent Data Management (IDM) research effort which has, as one of its components, the development of an Intelligent User Interface (IUI). The intent of the IUI is to develop a friendly and intelligent user interface service based on expert systems and natural language processing technologies. The purpose of such a service is to support the large number of potential scientific and engineering users that presently have need of space and land-related research and technical data, but have little or no experience in query languages or understanding of the information content or architecture of the databases of interest.

This technical memorandum presents the design concepts, development approach and evaluation of the performance of a prototype Intelligent User Interface (IUI) system for the Crustal Dynamics Project Database, which was developed by the Crustal Project and is managed by the NSSDC. The Crustal Dynamics IUI was developed using a microcomputer-based expert system tool (M. 1), the natural language query processor (THEMIS), and the graphics software system (GSS). The IUI design is based on a multiple view representation of a database from both the user and database perspective, with intelligent processes to translate between the views.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Intelligent user interface, expert systems, natural language processing, databases	Unclassified-Unlimited Subject Category: 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	55	

NASA FORM 1626 OCT 86